A behaviour driven recommender system in the fashion domain.

R.B.G. Starmans

Technische Universiteit Delft

Delft University of Technology

Challenge the future

A behaviour driven recommender system in the fashion domain.

by

R.B.G. Starmans

to obtain the degree of Master of Science
in Computer Science
at the Delft University of Technology,
to be defended publicly on Wednesday June 19, 2019 at 13:30.

Student number:	4141792	
Thesis committee:		
Chair:	Prof. A. Hanjalic,	TU Delft
University Supervisor:	dr. C.C.S. Liem,	TU Delft
Committee Member:	dr. ir. A. Bozzon,	TU Delft
Company Supervisor:	S. ten Dam,	Sanoma, Fashionchick
Company Supervisor:	D. Timmers,	Sanoma

Abstract

Web shops use recommender systems to help users find the products they find interesting in the large amount of available products online. An often used approach to do so is collaborative filtering. This method relies on historical user-item interactions and uses them to recommends products other users found interesting. Fashion is very reliant on quickly changing trends and personal preferences and requires a more personal and up-to-date approach. The focus of this research is to generate recommendations based on what products the user is currently searching for. It does this by detecting user behaviour based on the search scope of users and products user look at in the current session. Then new products are recommended by means of clustering new products to the most interesting products of the current session. This system was then compared with item-based collaborative filtering with an A/B test on the fashion platform Fashionchick.nl. It was found that traditional collaborative filtering was slightly more effective, but because of the small differences it is concluded that a behaviour driven recommender system are be promising and that more work is needed.

Preface

This thesis is the last task I have to complete for my Master of Science degree in Computer Science at the Delft University of Technology. During the last year I faced a lot of challenges, but now I can proudly present to you my completed work. I learned a lot, but this would not have been the same without the support of numerous people. First and foremost I would like to thank Cynthia Liem for supporting me during this time as my supervisor and for her ideas and advice. Next, I would like to thank all the people at Fashionchick. My time at Sanoma would not have been the same without you. I would like to thank Sjoerd & Sander for supporting me and acting as a guide within Sanoma. From the Data Science team at Sanoma I would like to thank Dennis for his help and feedback on the implementation of my system. I would also like to thank my family for always supporting me during my time as a student. It has not always been easy and without you I could not have done it. A special thanks goes out to Anouk, my girlfriend, who has been a huge support to me. She kept me going during the most difficult moments. Lastly, I would like to thank the other members of my thesis committee, Alan & Alessandro for reviewing my work.

R.B.G. Starmans
Delft, June 2019

Contents

List of Figures

List of Tables

1 Introduction

In the field of recommender systems a lot of research has happened since the rise of the web. Ever since Netflix organised a competition to improve their recommender system in 2006 the field has been booming with new developments [10]. The goal is to quickly get the most relevant items to users to keep them attracted to the platform. This is no different in E-commerce. Practically every webshop you can find issues some form of recommender system to get the users in touch with products they might find interesting. With the ever growing E-commerce market [13], it is a hot domain for researchers to explore.

There are multiple approaches that are often used in webshops to recommend products to users. The most well known is of course the “others also viewed” section on a product page, which employs item-based collaborative filtering. This serves the user other products that are relevant to the one that is being viewed. Other, more personal, approaches also use collaborative filtering, but these do not try to find other similar items, but rather other similar users, and other relevant items based on those users, this is user-based collaborative filtering. Collaborative filtering uses a User-Item Matrix, which consists of user-item interactions, usually ratings, and thus contains interest profiles for users and for items [57].

Recommender systems are deployed in all kinds of domains and each recommender system is used for different goal, which depends on the domain. An entertainment platform such as Netflix has a subscription based business model, and thus their goal is to make sure users like the service and in extension of that retain their subscription [34]. An E-commerce platform such as Zalando has the goal to make users buy more products by helping them find the products they are interested in [42]. These different goals lead to different choices during the design of the recommender system.

1.1. Recommender systems for fashion

In the fashion domain recommender systems are also widely used. Fashion is, however, a very different product than most other products. First of all, fashion is subject to trends. The styles of clothing that are popular right now, might not be anymore in a few months, or even weeks. While popularity is something to take into account in other domains as well, in fashion it might behave differently because of the quickly changing trends. This is something that needs to be kept in mind with recommending interesting products to users. When something stops being a trend, it usually is because something else has replaced it. With recommender systems you want to be as close to those trends as possible.

Related to trends are the seasonal trends; these change not only on what is currently popular, but rather on the time of the year. A winter coat is not very useful during summer and a swimsuit is not useful in winter. Furthermore, there is the personal preference of users. Besides the fact that people do follow trends, they also have their own style that they feel comfortable with. People normally even have multiple styles that they like. For example, one person can be shopping for one style of clothing for work, as well as a more casual style for their private time. In addition to multiple preferences a user can have, a personal preference is also subject to change. It can be both influenced by the current trends as well as simply change over time.

Therefore, building a recommender system in a fashion domain might require a more personal and short-lived approach than traditional collaborative filtering. While collaborative filtering generally is very successful in recommending relevant products to users, it still uses preferences of other users. This mostly works for general fashion trends, but does not take personal preferences of users, or even current shopping motivations, into account. Other, more temporal, approaches could perform better for such a complex and quickly changing product as fashion.

1.2. Research objectives

Making product recommendations in E-Commerce happens everywhere, based on different data and with different methods. The fashion domain is no different in this. However, fashion is personal and trends pass more quickly than for most other types of products. In this research, the focus will be on a recommender system tailored to the user based on live information of the user's behaviour on the website. The theory is that users visit a webshop with a certain buying intention, which translates to their behaviour, which in turn can be used for recommendations. The main research question then becomes:

Main Research Question: *How well does a behaviour driven recommender system perform as an alternative to traditional item-based collaborative filtering in the fashion domain?*

The answer to this question can be found by implementing a full working system that captures the behaviour and interests of each user, and uses this information to generate recommendations while the user is browsing the website. For such a system to succeed, we first want to find out how to detect user behaviour and if certain patterns in this behaviour can be found. We will investigate this with the following research question:

RQ1: *How can browsing behaviour best be categorised and what shopping patterns can be discerned?*

To answer this question, we will analyse user behaviour with a set of relevant features and divide the users into groups based on those features. These patterns or shopping modes will be used to generate matching recommendations and it is important to test how effective these recommendations are. For this we propose the following research question:

RQ2: *How effective is recommending products based on the user's browsing behaviour?*

This question covers the behaviour driven recommender system in general, but also the approaches to distinguish the browsing patterns. We want to see if the chosen methods work for the matching user behaviour. For example, for one type of behaviour, the corresponding recommendations might perform well, while for another it might not. This question will give insight into that.

Finally, we want to assess the system as a whole. Making recommendations in real-time, and based on information of a user's live browsing session, poses limitations on the system because of data sparsity and time constraints. Recommendations should be ready before the user visits another relevant page, but should also stay up to date with the subsequent user behaviour. To test the ability of the system to do so, we analyse it with the last research question:

RQ3: *How feasible is a session based approach with real-time generation of recommendations?*

1.3. Outline

In this thesis we will first talk about previous work that happend in relevant fields to this research in chapter 2. We will cover session modeling, user behaviour and intent, recommender systems, the fashion domain and then the evalutation of recommender systems. In chapter 3, we will outline our methodology to analyse user behaviour and interests and the implementation for a behaviour driven recommender system. After that in chapter 4, we will talk about our use case Fashionchick, where we will implement and test the system. In chapter 5, we will discuss the analysis on data from Fashionchick which will be used in the online system. The full implementation of the online system will be discussed in chapter 6 and after that the experimental setup to test the system will be discussed in chapter 7. Lastly the results and the conclusion will be covered in chapters 8 and 9.

2
Previous work

In this chapter, the main areas of interest for this research will be covered. First, we will cover the modelling of user sessions, as this forms a basis for the data input of the recommender system. Then, we will go into the capturing of behaviour and intent of users online. After that, recommendation algorithms, techniques and intent specific recommender systems will be detailed. Lastly, the evaluation of a recommender system will be discussed, in offline as well as online testing.

2.1. Session modelling

In an early paper from 1995 a user session was defined as “..all of the page accesses that occur during a single visit to a Web site” [27]. A typical website visit consists of a user opening the website and visiting a few pages, after which the user has found what was needed and closes it again. This collection of page visits would accumulate to a single session. In [27], the first proposal was to gather the data about a session from raw server logs. Now, a general approach to collect data about a session is to register click events to objects on the web pages, and monitor the session of a user with the clickstream that is generated from these events [18, 49, 50]. This method allows the developer to determine what information is gathered about the session and when, which makes the process much easier. Click events can be accompanied with data about what the page was, what information was on it and what item was clicked on. This makes click events a very effective method for data gathering.

These clickstreams give an idea of what the user is doing on the website during a session. But users often close the page, effectively ending the session and then reopen another page of the website again. Different users can be identified by using their ip address for example, but when users visit the website multiple times within a certain time frame these visits should be identified as different sessions at some point. There are different approaches to accomplish this [3, 24, 48], but a typical method is using a timeout set at around 30 minutes.

A model to capture user visits to a website is proposed in [31]. With this observation module, as it is called, the complete visit is captured, with the sequence of pages that are visited and which links the user followed. This module also contains information about what products are added to the shopping cart, as well as past purchasing behaviour. In this way, a broad representation of a user’s browsing session is constructed. This gives insight in their behaviour and why certain choices are made. More about how this information is used to predict the user’s intent will be discussed in section 2.2.

In [50], a system to model web browsing using clickstream data is proposed, and this is then used to predict a user’s navigational path and if that user is likely to make a purchase. First, a Markov Model denoting the type of page the user is looking at, is used to construct the browsing path. The type of page and the order the user looks at those pages is an important part of the session. Data about all the transition between the pages is used to make a transition matrix and to calculate the probabilities of the transitions from each page type to another page type. Then, a multinomial probit model was used to monitor covariates that might have an effect on the path the user follows.

Another approach using clickstream data to construct the browsing path of the user is taken in [56]. First, a website topology is constructed which can be seen in figure 2.1. It consists of a tree where the homepage is the root, then main categories are children of the root, which are in turn parents of subcategories and lastly the items in the categories are represented by leaf nodes. The browsing path is then constructed using the depth of the node corresponding to the page in the website topology. An example browsing path is: $\{Homepage, category_2, category_{21}, item_1^2, item_2^2\}$. The advantage of this approach is that it contains a lot of extra information about what the user is actually viewing. The browsing path is augmented with the depth the user is visiting the website.

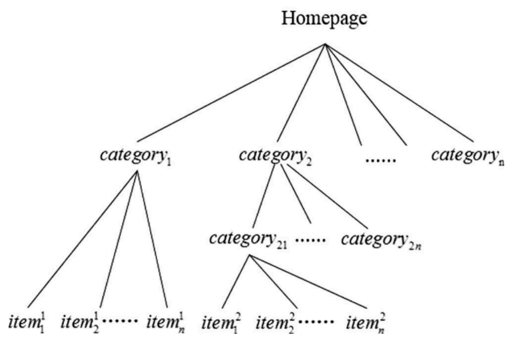

Homepage
$category_1$
$category_2$
......
$category_n$
$category_{21}$ ...... $category_{2n}$
$item_1^1$ $item_2^1$ ...... $item_n^1$ $item_1^2$ $item_2^2$ ...... $item_n^2$

Figure 2.1: E-commerce website topology [56]

In [47], sessions are also modelled with the browsing path. Here, the path is represented as a sequence of items the user views with an encoder-decoder structure, first proposed for statistical machine translation in [25]. The sequence is encoded with a bidirectional RNN, forwards and backwards. The encoded sequence is represented by two vectors of the hidden states that are generated, these vectors are of fixed length. From this sequence, predictions about purchases are made. More on that in section 2.2.2.

2.2. Capturing user behaviour & intent

In this section, capturing behaviour and intent from a user's browsing session is covered, as a session contains a lot of useful information about what is being viewed, in what order and what items might be interesting to the user. This could be a good basis for capturing the behaviour and intent of a user. To learn a bit more about user intent, we first look at intentions users might have behind search queries. In the optimisation of search engine results the goal of a query proved to be very important.

2.2.1. Query intent

In search of why users want to visit a certain website, in [21], three main goal categories for queries users enter in search engines are proposed: Navigational, Informational and Transactional. In the navigational category a user wants to access a certain website, in the informational category a user wants to find information about a certain topic and in the transactional category a user wants to perform further actions on a website. In a webshop, it could be possible to see similar categories of intent. For example, people could be trying to find information about a product. Also, a person that wants to buy a product could be linked to the transactional category.

A way to automatically capture a query goal is to analyse historic user click behaviour. From this, the distribution of the clicks over the results to the same query can give insight into what the corresponding goal might be [45].

Because the use cases for a search engine are so different from that of a shopping platform, queries might not be as useful to capture shopping intent. When the user enters a query to search in a shop, he or she is most likely to find a product or category in that shop. However, some lessons can still be learned from it. An analysis about query classification happens in

[19] where queries are represented by words from the relevant search results. This content based approach is an interesting way to capture information about the results that were interesting to a query, and can also be used to capture information about interesting items that a user viewed in a webshop.

2.2.2. Behaviour & intent

The browsing path a user takes in a session is a useful method to gather information about his or her behaviour and this behaviour can give insights into why the user is visiting a website. If the user has the goal to purchase a certain product, this user is probably looking at specific occurrences of this product, to compare products and decide which one to buy. Such a session can be easily distinguished from a session in which a user is just looking randomly at all kinds of products. In the latter case, the goal or intent is much less apparent. By monitoring the browsing paths and behaviours of users, their intent can be extracted or predicted [31, 47, 56]. In this section the capturing of behaviour and extracting user intent from that behaviour will be covered.

The study of user behaviour has a strong basis in psychology. Just as the research done in [30], which based its theory on the S-O-R model from [47]. Here it is said that the stimulus S has an effect on the state of the organism O, which in turn determines the response R. This can also be applied with shopping behaviour because a website shows certain products or stimuli, which then influence what the user thinks about the website and that determines their behaviour after that. So what is done in [31], is to model the user intent in real time and how it changed over time. This was new, as previous papers such as [50] made the assumption that the intention would not change during a session. A Hidden Markov Model is proposed that has three parts: the starting probabilities of the intent states, the transition probability matrix and the waiting times of the intent states [31]. These items are used to predict what the next intent state is going to be, where the waiting times represent how long a user stays within a certain intent state.

While the proposed Hidden Markov Model based on the browsing path in [50] is ultimately used to predict whether or not a user is going to make a purchase, the browsing paths can also be used to determine other forms of intent which are not necessarily related to purchasing. A good example of that is performed in [49] where clickstream patterns are used to determine what kind of shoppers are visiting the website. First, four shopping strategies are identified, namely “directed buying, search/deliberation, hedonic browsing and knowledge building”. Then, a set of metrics about the shopping session is set up. These metrics consist of information such as the percentage of pages that are home, search, category, brand or product pages. But also metrics that indicate the variety of category, brand or product pages, measured by the number of pages that is unique. Then, these metrics are linked to expected patterns of navigation by the four shopping strategies. These four strategies combined with the expected patterns then represent a certain form of behaviour that a user is likely to exhibit when following a certain strategy. The table that is created for this can be seen in table 2.1. For example, a user who is directed at buying, is likely to view more product pages than category listings, while a user who is just browsing for fun will likely visit more category pages and the variety of categories that is visited will be also higher.

	Focus of Session	Category Variety	Product Variety	Repeat Product Viewings
Directed buying	Product pages	Low	Low	High
Search/deliberation	Category, product pages	Low	High	Moderate
Hedonic browsing	Category pages	High	High	Low
Knowledge building	Information pages	Low	Low	Low

Table 2.1: Expected Pattern by Shopping Strategy [49]

As discussed in section 2.1, in [47], sessions were also modelled using the browsing path and used in an encoder-decoder structure. With the forward and backwards encoded vectors they predict whether or not one of the items a user visited is likely to be purchased or not. In this article only two classes were identified, purchase and browsing, which were modelled as a binary variable.

2.3. Recommender systems

The internet explodes with content. Websites such as YouTube accumulate content so quickly, that in 2013 alone, 100 hours of video were uploaded every minute. In 2014 it already increased to 300 hours of video uploaded every minute [15]. Users cannot keep up with the amount of content that is added so incredibly fast, and that is why recommender systems have been popping up everywhere. In 2006 Netflix even issued a 1 million dollar prize to improve their recommender system [10].

Nowadays recommender systems come in various forms and implementations, but for the most part can be categorised in the following categories: content-based recommender systems, collaborative recommender systems and hybrid forms of recommender systems [16, 18, 37]. Content-based recommender systems use item similarity to recommend a user new items based on items that were liked before. Collaborative recommender systems use historic user preferences to recommend new items to a user. Hybrid forms of recommender systems use a mix of the first two methods.

2.3.1. Content-based recommendations

As mentioned in the introduction of this section, content-based recommendation uses similarity of items to items previously liked by the user to recommend new ones. To enable this to work, the following things are needed: some measure of characterisation of the item, followed by a way to compare two items based on this characterisation.

The internet is full of articles and documents consisting of text. Comparing items based on the text that is contained is a frequently used approach. Term Frequency-Inverse Document Frequency [54], or TF-IDF, is a statistical method to determine the importance of a word for a document in a result list of documents. TF-IDF for a word can be computed by the product of the term frequency by the inverse documents frequency:

$$TF \cdot IDF = f_{t,d} \cdot \log \frac{|D|}{|\{d \in D | t \in d\}|} \tag{2.1}$$

Where f is the frequency for term t in document d and D is the result set of documents. Based on the importance of the words, two documents can be compared in similarity by putting the TF-IDF for the n most relevant words in a vector and computing the distance between those vectors.

To calculate the distance between two vectors multiple approaches are possible. Two widely used methods are Euclidean distance and cosine similarity [44, 52]. For items a and b with dimensions n, Euclidean distance d_e is defined as:

$$d_e(a,b) = \sqrt{(a_1 - b_1)^2 + (a_2 - b_2)^2 + \cdots + (a_n - b_n)^2} \tag{2.2}$$

The cosine distance d_c between items a and b with dimensions n is defined as:

$$d_c(a,b) = 1 - \frac{\sum a_i \cdot b_i}{\sqrt{\sum a_i}\sqrt{\sum b_i}} \forall i \in \{1,n\} \tag{2.3}$$

Both distance definitions are taken from [44], and are adapted for the items dimensions to be n. Recommendations can be made to the user by comparing previous documents the user liked with new documents based on TF-IDF and distance measures.

When an item is not a document represented by text, other features to identify the item have to be extracted. This is most simple when these features can be extracted automatically [16].

2.3.2. Collaborative recommendations

Collaborative recommender systems recommend items based on historical data about what other users have liked. For this, generally a User-Item Matrix is used to store ratings users give to items, an example can be seen in table 2.2. Different scales of ratings are possible. Usually ratings happen on a Likert scale which normally ranges from 2 up to 10. Other scales are also possible but usage of those can depend on the use case. Netflix currently uses a two point Likert scale to rate items [9]. Which is most likely due to it taking less time to perform for a user than deciding what grade to give it on a scale of 1-10. Also methods which require less interaction of the user exist, such as implicit unary ratings [41]. For a webshop this could be inferred from viewing a product page or purchase of an item. However, because these ratings are implicit, there is no explicit indication if and when the user dislikes an item.

User/Item	i_1	i_2	i_3	i_4
u_1	1		1	1
u_2	1	1		0
u_3		1	0	1
u_4	0	1	0	

Table 2.2: Example User-Item Matrix with binary ratings.

Collaborative filtering algorithms are implemented in two ways, memory-based or model-based [16, 20]. Where memory-based algorithms use the User-Item matrix directly and model-based algorithms use it to train a model offline, which is used for the recommendations.

Memory-based collaborative filtering

First, we will go over memory-based collaborative filtering [20, 29, 55]. As the User-Item Matrix has two dimensions this can be interpreted in two ways. The first way to predict new items is based on the user. From the user model with corresponding ratings for items, other similar users are found by using cosine similarity, as discussed earlier in equation 2.3, or correlation between rating vectors. The Pearson Correlation pc for user a and b, where n is the number of ratings, can be calculated like this [16, 53]:

$$pc(a,b) = \frac{\sum(a_i - \bar{a})(b_i - \bar{b})}{\sqrt{\sum(a_i - \bar{a})^2(b_i - \bar{b})^2}} \forall i \in \{1, n\} \tag{2.4}$$

Memory-based collaborative filtering can also happen through items instead of users. This is a method often used when a user is looking at a certain item and there is a list of other interesting items displayed. These are items that other users often find interesting as well. So similar items based on ratings. These similar items are also found by means of correlation or cosine similarity with rating vectors as input.

When the most similar users are found, the new items to recommend are found by selecting items these similar users liked. When the most similar items are used, other items liked by users that liked the current item are recommended. Items that the current user already rated are excluded.

Model-based collaborative filtering

Next is model-based collaborative filtering. In this method a model is trained offline which is used to recommend items [16, 20]. A widely used approach is Matrix Factorization [20, 43]. The idea behind this method is that the User-Item Matrix can get really large, so a matrix factorisation is used which results in lower dimensional matrices which are then used to recommend interesting items. The matrix factorisation is based on the singular value decomposition of [33] and looks like this:

$$UI = U\Sigma I^T \tag{2.5}$$

Where UI is the User-Item Matrix, U the User Matrix and I the Item Matrix with lower dimensions and Σ is the Singular Value Matrix. U contains vectors about how each user typically rates items, I contains vectors about how items are typically rated by users. Dimensions of vectors in U and I are the same [33], and each entry represents a latent factor and how the user or the item scores on that factor. The Σ Matrix is a matrix with the same size of UI and has only nonnegative real entries on its diagonal [33]. This diagonal represents the singular values of UI and translates the weight of each latent factor from user to item.

This matrix factorisation model can be trained offline and then a rating r, for an item i, that a user u might give can be estimated by the dot product of the factorisation for that user and that item like this:

$$r_{ui} = U_u \Sigma_{ui} I_i^T \tag{2.6}$$

2.3.3. Behavioural and intent-based recommendation

Recommender systems traditionally use historical user information in which users handled explicitly. For example, historical user information such as ratings from the User-Item Matrix, purchases that were made previously or items that other users looked at for item-based collaborative filtering. Behavioural recommendations do not necessarily focus on this, but rather on the current behaviour that users exhibit during the session.

In [17], an implicit user feedback model to re-rank query results is made. This feedback model consists of a set of user actions which represent post-search navigation history. These user actions consist of features from clickthrough information, such as position of the item that is clicked or frequency that the link is clicked. In addition to that user actions can also be browsing features such as time on page, the number of clicks from the search page, or average dwell time on the page for the current query. With this set of features that is collected from the navigation history the result list is re-ranked in different ways. The features can be used to directly re-rank the result list, also a neural net ranker is used to learn weights for all the features.

Another intent-based recommender system is proposed in [37]. Based on a Hidden Markov Model built from clickstream data, it is inferred how likely a customer is to buy an item. Based on how strong this likeliness is, a different approach is taken to recommend items. This ranges from showing the top 10 products of the current category, to items currently in the shopping basket or even personalised discount offers. Which of these methods is used is determined based on business rules. Interesting about this is that there is not one single method applied to all users. This seems like a very logical step as users with different goals might benefit from different approaches. Another important lesson is that the likeliness that a user is going to buy can change during a session, which is an important concept also seen in [31].

In [40, 42], the relevance of certain types of items that could be relevant during the current session of a user are researched. The research happened with a dataset of Zalando and a test was run with small adaptations to the recommendations of the website. The importance of short term user intents, items that the user has already seen, discounted items and the effect of popularity trends in fashion are compared. For the short term intent, it was found that the average shopping session had 9 visits to different item pages, which in turn belong to on average 2.7 different categories. With around 330 categories, this seemed to indicate that users mostly had a certain direction in their navigation and thus a certain shopping intent. Moreover, it was found that after users looked at a product and then an item of the same brand, price segment, category or color was shown, there was a significant increase in the conversion rate [40, 42].

2.3.4. Result diversification

Earlier it was already seen in [49] that the shopping strategy of a user can be detected. As each shopping strategy has a different focus, it might also need a different variety of recommendations. Someone who is looking at a low variety of categories probably does not want to see products from a wide range of categories he or she did not even look at. For that, result diversification is a technique that might be useful in adjusting results according to the shopping behaviour of the user.

In [36], three result diversification techniques were tried to make the results for queries more relevant to the intent behind it. While the techniques are directed at query results, one of them, Single Pass Clustering, seems interesting as clustering can happen on other characteristics than just text. First introduced in [39], this clustering method works by comparing documents against each current cluster. In [36], this happens by means of TF-IDF. When a certain threshold is not met for any cluster, then the document is assigned to a new cluster. Lastly, [36] outputs the highest ranked documents of each cluster to output for the results. This method is useful as it is able to quickly identify clusters in a set of documents and give a diverse look into the result set.

2.4. The fashion domain

The domains in recommender systems are widespread from search engine results for Google, E-commerce webshops such as Zalando or Amazon, but also media platforms such as YouTube or Netflix. Most approaches have been tested in multiple, if not all, areas. As the results of a recommender system depend on the data that is used, it is good to keep in mind what underlying characteristics the data possesses. Fashion is a domain that is different from other E-commerce domains.

The temporal factor is very important in the fashion domain. In [18] it is mentioned that products in the fashion domain are very short-lived and trends change quickly which results in sparse transactional data for these products. It is thus not surprising that approaches for recommender systems in fashion often rely on the features of the products [18, 38]. Then, the sparse information that exists about what the user is looking for, can be used to gather other products that the user might find interesting. This can be done in a content-based way.

Something else to keep in mind with fashion is what is described in [30]. In this article it is described that users not only have one single personal preference, but users can have multiple personal preferences for different styles. This is something that was already mentioned in the introduction in section 1: different occasions, such as casual and work, require different styles, in which users, again, have different preferences. Users can be shopping for these different styles over the course of multiple sessions as well as during a single session. This is important to detect, and respond to with recommendations.

2.5. Evaluation

The success of recommender systems relies on the quality of the results of the recommendations. This is measured in how well the recommended items actually fit the user's needs and interests. In order to test what items are good and which are bad, there need to be certain metrics to measure what the user thinks about them. With these metrics the recommenders systems can be compared as to how good the items fit the user's needs and interests.. This comparison can happen online and offline. Where an online system actually performs recommendations, serves them to the user and tests how the user responds to the recommendations. An offline system only uses historical data of how users interacted with items and recommends items to users within the space of interactions that already exist. Both approaches have their advantages and disadvantages. Offline testing is good to quickly test multiple recommendation approaches as it happens offline, and thus does not need a fully developed and running system. However, it happens on historical data and as already mentioned, it can only recommend items from the set of items the user has already interacted with. While in online testing any item can be recommended and an actual user response to it can be recorded. It does however need a fully working system to do so. When making changes to this system, it also directly changes what the user gets to see, so there are limitations to what you can show to the user. For example you cannot simply show the user two sets of recommendations and let them interact with both of them.

2.5.1. Offline testing

For offline testing, a data set in which a set of users have rated a set of items is needed. This data is then often split into a training and a testing set to prevent overfitting. Then multiple recommender systems are compared on the same data set for better comparison. Two often used metrics for this are precision and recall from information retrieval [22, 28]. Precision and Recall are defined as follows:

$$\text{precision} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}} \tag{2.7}$$

$$\text{recall} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}} \tag{2.8}$$

Precision represents the fraction of retrieved items that are relevant to the user and recall represents the fraction of relevant items that is returned [22, 28].

For recommender systems, precision represents the fraction of recommended items that are relevant to the user, and recall represents the fraction of relevant items that is recommended to the user. These metrics cannot be used in an online test as not all information for them is available.

2.5.2. Online testing

Recommender systems are also tested online to test how a system works in reality, with real users. For this a fully implemented system is required, which can then be tested by means of an A/B test. This method is normally used to test a new method against a control group. Testing with this kind of test also takes longer than an offline test as the test has to be live for a certain amount of time. First of all, there needs to be a certain amount of data points for the measured difference to be significant, this depends on the number of users that are in each test. Secondly, websites typically have different patterns of how many users access it each day. These users can also behave differently each day. For example users can be more eager to purchase items on a Friday afternoon because of the weekend. These are all things to take into account when running an A/B test.

Metrics covered in the previous section, section 2.5.1, are not useful here as the ratings that are needed are not always available, as items can be recommended that the user did not rate yet. Because of that, other metrics need to be determined carefully. These other metrics could technically be anything. Netflix is a great example for recommender systems, and in [34] it is detailed how metrics are chosen for the A/B testing. It is explained that Netflix is a subscription based service and that thus the most important goal is to keep users subscribed. This is called the retention of users. They also mention that the retention rate is strongly correlated with the viewing time, so increasing the viewing time normally means a higher retention rate.

Webshops such as Zalando also run these A/B tests, but with a different business model than Netflix, they use different metrics. In the end they want to increase their sales, so the metrics for this then is the conversion rate of users into buyers. They also look at other metrics such as click-through rate, absolute number of sales and the generated revenue [32]. Depending on what the business thinks is important for them to improve, these metrics can be adapted.

3 Methodology

In this chapter, the methodology to build a behaviour driven recommender system in the fashion domain will be detailed. First, the general use-case of a fashion web shop recommender system will be discussed, as well as the motivation why this approach is taken. After that, the techniques that will be used to build this recommender system will be discussed.

3.1. Use case: Fashion web shop recommender system

Online webshops have the goal to make the user purchase as many products as possible. An important step to do so is to make the whole shopping process as easy as possible for the user. Easy site navigation with a proper product search engine are key in helping the user find what they are looking for. With the growth of web shops everywhere came an increased number of products, for which web shops applied recommender systems to give users recommendations of products which might be interesting to them. One of the major approaches in recommender systems on web shops is the ”others also viewed” section on a product detail page. This section works with item-to-item collaborative filtering. So for each item in the shop a list of recommendations is determined. An explanation of this method can be found in 2.3.2. The input data is a table with an entry for what products each user has viewed, called the User-Item Matrix. The list of products that gets recommended next to a product is a list of products that other users most often also view when the current product is viewed. Because of this the recommendations might cater to the user because of a general interest pattern that is common across a majority of users. However, a domain as personal and temporal as fashion might require a different approach that is focused on the user and their intentions while they are shopping. So instead of the collaborative filtering approach of recommendations in a fashion webshop, this research applies a different approach to recommend products to the user, which is based on the shopping behaviour during a session and the interests of the user during that session.

Web shops often track their users around the website with clicktrackers. The data gathered by this is a basis for the recommender system, as it contains information about what users look at, what they buy but also what products are of similar interest to users. This clicktracker data typically looks like, or is at least similar to, this:

date	page	category	price	brand	material	color
13:00:50	category	Clothing				
13:01:45	product	Clothing	49.99	Brand A	Cotton	Gray
13:02:00	category	Shoes				
13:02:39	product	Shoes	79.99	Brand B	Leather	Brown
13:12:08	category	Accessories				

Table 3.1: Example of how clicktracker data typically is structured.

Contained in this data is a lot of useful information about shopping behaviour of the users.

It contains how the user is looking for a product, what the scope of their search is and also what types of products the user is looking at. As a web shop, this is information you want to utilise. Especially considering the goal of a web shop, which is to make sure the user finds what they are looking for. With an insight in these patterns there is a strong potential for a more personal and suitable recommender system to be made.

3.2. User behaviour & interests

To make useful recommendations to the users based on the information contained in the data as structured as in 3.1, there are 2 main things that we need: the behaviour of a user and their interests during a single visit. The behaviour is used to determine the approach of recommendations that will be taken and the interests for products as a starting point to recommend new products. For this, we will analyse an offline collected set of data in which users are identified with a user identifier, where sessions are represented by a set of events within a single visit split by at most 30 minutes of inactivity.

3.2.1. Feature selection for user behaviour

For the analysis of the sessions and to see whether or not certain behaviour is even present in users visiting a web shop, it is determined what to look for in the available data. Different approaches to determining behaviour and intent are considered.

Because click and pageview events are timestamped to easily keep track of when they happened, the consideration of constructing Markov Chains was made. Markov Chains are found to be useful to predict specific intentions of the users on the website. As we have seen in section 2.2.2 with the research from [50], it was seen that it was possible to predict how likely a user is to make a purchase. This however does not match the goal of this research, which is not to make a prediction about such a specific intent but rather to capture a more general shopping behaviour.

To capture such shopping behaviour on the web shop, features from [49] were selected as the goal of that paper was more in line with the research goals of this thesis. These features are category variety (CV), product variety (PV) and repeat product viewings (RPV). We define a session S, a set of pageviews with categories C, and a set of products pageviews P, the features to capture behaviour are then defined as follows:

$$CV = \begin{cases} 0, & \text{if } |C| = 0 \\ \frac{|unique(C)|}{|C|} & \text{otherwise} \end{cases} \tag{3.1}$$

$$PV = \begin{cases} 0, & \text{if } |P| = 0 \\ \frac{|unique(P)|}{|P|} & \text{otherwise} \end{cases} \tag{3.2}$$

$$RPV = max(|p|) \quad \forall p \in P \tag{3.3}$$

Important to note about the sets of pageviews P and C is that for each session holds: $P \subseteq C$, because product pages also have a category. Another thing to note, about CV, is that webshops generally have multiple levels of categories, with main categories and sub categories. There is only 1 feature for the category variety, so these levels have to be combined into this 1 feature. This can happen for example by calculating the feature for each level and then taking the average over these 3 levels.

These 3 features, CV, PV and RPV, tell us something about how the user is browsing the website. It shows how deep or shallow the user is looking at category pages and product pages, and how directed a user is browsing. These features are not reliant on how they are ordered which makes them easy to extract from the data. Ordering of the data will become more important when streaming events online but this will be discussed further in chapter 6.

3.2.2. Detecting shopping modes

With these features CV, PV and RPV, we want to see whether or not there are certain patterns in the behaviour of the users. To see if there are any general patterns just as was seen before in [49] in section 2.2.2, there were 4 major shopping strategies, Directed buying, Search/Deliberation, Hedonic browsing and Knowledge building, each with its own pattern in the selected features. This analysis was done for a nutrition web shop which is a vastly different domain than fashion, so the patterns might also differ, but seem general enough to serve as a starting point.

This pattern analysis will happen by means of clustering the sessions together based on the given behavioural features with the k-means clustering algorithm [35, 46]. This clustering algorithm will choose k random sessions as centre points, where the point values are represented by the selected feature values CV, PV and RPV for that session. It will then cluster the remaining sessions to the centre points by means of euclidean distance to the centre points. A session is assigned to the cluster to which it has the smallest distance. When all sessions are assigned, the centre points of each cluster are recalculated. This happens by taking the average of the points in each cluster. Then the clustering starts over. The algorithm will iteratively improve the centre points until the difference between the centre points of 2 iterations is smaller than a preset threshold value. The output of the algorithm are the centre points of each cluster. In the next chapter, chapter 5, we will go over how to choose k. This will influence how many clusters there are. First we want to see how the patterns match with the shopping strategies from [49]. The different shopping patterns will later be used to determine how products will be recommended, which fill be further discussed in section 3.3.3.

3.2.3. User interests & product relevance

To recommend products to a user based on the information that is available about them during the current session, we not only want to capture a general behaviour of a user but also the products a user looks at, and which of those are the most interesting. We then get contemporary interests in products for each user.

While it is impossible to look into the mind of the users and see how interesting a product is to them, we have to work with the information that we do have. Essentially, we want to measure the user interaction or engagement with the product page. In the event data that is available some information is embedded to possibly get an insight into this interaction. For this we will look at the following set of features:

- The time spent on the product page: tpp
- The time spent in relevant boundaries on the product page.: $rtpp$
- The amount of scroll activity on the product page: scr
- The number of pageviews for that product: npp
- A custom sequence score, based on the order the products are viewed in: seq
- A custom sequence score, based on the order the products are viewed in, combined with relevant time on the page: $rtseq$

The npp and the scr features are simple. scr is a field in the events so can be easily extracted and the npp is simply counted. The tpp feature is calculated by taking the difference between the timestamps of the pageview and the pageview that comes after it. $rtpp$ is a scoring feature which appoints a score as follows:

$$rtpp = \begin{cases} 1, & \text{if } tpp \geq 5 \\ 0 & \text{otherwise} \end{cases} \tag{3.4}$$

Where less than 5 seconds on the page is deemed to be too short and means that the user is not interested, also called a bounce. The seq feature is a bit more complicated. Given a category page containing a set of products P, if a user will click on 1 or more products, the first will be likely be the most interesting and products after that will be deemed less interesting. The sequence score seq for a product p_i is calculated as follows:

$$seq_{p_i} = \frac{1}{i} \tag{3.5}$$

In this equation i starts at 1 and is incremented each time the user clicks on a product. When the user visits a new category page i is reset to 1. The last feature, $rtseq$ combines the seq and the $rtpp$ features, and is the same as the sequence score but appoints a score of 0 when the time on the product page is below 5 seconds.

To find out which feature works best to determine user interest in products, the correlation score between the features and the conversion rate for a product is calculated. The best scoring feature will then be used in the online implementation. Important to note, is that all features will be calculated per product and not per session and then per product.

3.3. Live behaviour based product recommendations

Making a recommender system driven based on behavioural browsing information of the user at that moment, has certain implications to the design specifications of that system. Collecting data about the user in real time, consecutively making recommendations which that user will get to see on a next pageview, poses certain limitations. The most important constraint is of course time. As described, the recommendations have to be finished and ready to be sent back to the user as soon as possible. So that when a user visits another page, the recommendations can be shown. Another constraint is the available data about a single user session. With an average of 9 product pageviews for the dataset of Zalando in [42], it shows how limited the data collected about a session can be. With that in mind, the challenge stands to get as good as possible recommendations out as fast as possible.

3.3.1. Online behaviour detection

With the preparation for user behaviour out of the way with the offline analysis on the click event data, the detection of behaviour in the live system can happen very quickly. The first thing that needs to happen is the collection of the same features CV, PV and RPV. As events enter the system gradually, in groups, and the features are calculated over the complete set of available events, the features will have to be recalculated every time a new event enters. This also holds for the detection of the behaviour for a session which is thus subject to change over time, just as was found in [31]. With the centre points for each cluster calculated as in section 3.2.2, the behaviour cluster a session belongs to can be detected by calculating the euclidean distance to each cluster. The cluster to which it has the smallest distance to is the behaviour cluster it belongs to. The centre point of this cluster will be used as a basis for the product ranking for that session. This will be discussed in the next section.

3.3.2. Online product relevance

After the offline feature selection process, described in section 3.2.3, we use the selected feature to calculate interest scores for each product a user looks at. So when an event enters the system that tells us that the user has looked at a product page, an interest score for that product will be calculated according to the selected feature. For each product a user looks at, this value will be stored once. Meaning if a user looks at a product twice, the score will be updated and only the newest value will be stored as it is the most recent information available. This results in a set of product identifiers with corresponding interest values which will be sorted based on the interest value. The top 1-5 products will be used as a basis for recommending other interesting products. How many products will be used will be determined by the user's behaviour which tells us how directed a user is at a certain product or product type.

3.3.3. Product recommendations

As previously discussed in section 1.2, the main goal of this research is to make a recommender system based on the behaviour of users and to test how well this works, and also compare the performance to traditional item-based collaborative filtering. Making recommendations based on the behaviour in a session is not something that has been widely researched before. Some methods were found, such as in [37], where different approaches were chosen based on business rules. While these rules are based on the user intent of how likely they are to make a purchase, there were no direct, automated links between their behaviour and the resulting recommendations. These links were found in the research of [40, 42], which had a focus on short and long term intent of users on Zalando and on how certain types of recommendations performed in those intents. Also a strong link was found between items users recently looked at and conversion rate when these users were shown similar items based on brand, price segment, category or color. This seems like a good basis to make recommendations for users on the time frame of 1 session. Then it can be monitored what the viewed items are and which seem to be the most interesting and use those to select other items.

The approach to the recommendations is to fit them to the level of direction the user shows in their behaviour. When a user, for example, seems to be very directed at 1 type of product, and thus has a lower level of PV but a higher level of RPV, we want to show only products of that same type. Whereas a user that has less direction at a certain type but has a higher level of CV we want to show a wider array of product types as well. To achieve this we will use clustering with a varying amount of clusters to cater to each group of user behaviour. First, we will discuss how product similarity will be calculated, which will then be used for the clustering discussed in section 3.3.5.

3.3.4. Product similarity

The next step to recommend relevant products to the user, is to find other products they might find interesting. Selecting these products will happen based on similarity between products interesting to the user. As found in [40, 42], the similarity features brand, price, category and colour are effective to select and rank products for recommendations. An attribute which was not mentioned is the material of a product but will also be used as a similarity feature because for fashion products the material can be very important. Especially with regard to user intent it can be important to use, as a user might be especially looking for a leather jacket instead of a synthetic jacket.

For a product p_i, we define a set of features. As mentioned earlier in section 3.2.1, web-shops generally have multiple levels of categories, for product similarity we only use the lowest known category level. In addition to that we will define the features brand, price, color and material as follows:

- The lowest known category level the product has: c_i.
- Brand of the product: b_i.
- Price of the product: pr_i.
- Color of the product: col_i.
- Material of the product: m_i.

The product similarity will be calculated as a sum of distance scores for each feature where a lower distance is higher similarity. For the brand, colour and material it is simple, and can be seen in equation 3.6, where f_i and f_j are the values for a feature for products i and j.

$$dist(f_i, f_j) = \begin{cases} 0, & \text{if } f_i == f_j \\ 1 & \text{otherwise} \end{cases} \tag{3.6}$$

The price can happen in multiple ways, by means of price range, as happened in [40, 42], or as a continuous measure. If used continuous, the distance between prices for 2 products will be calculated by the absolute difference, and can be seen in equation 3.7.

$$dist(p_i, p_j) = abs(p_i - p_j) \tag{3.7}$$

If the price is used with price ranges, each range gets assigned an integer and then distance is calculated the same as for the absolute difference. Which one of the two will be used is determined by analysing the distribution of product prices and ranges of products viewed in user sessions and will be detailed further in section 5.4.

The category distance is a bit more complicated as there are multiple levels of categories. We will work with 3 levels: main, sub and sub-sub categories. The approach is to think of the category levels as trees, where each main category is connected to the root, sub categories are children of the main categories and sub-sub categories are the leaves. For 2 products the distance between the categories is calculated as follows:

$$dist(c_i, c_j) = dist(c_i, lca) + dist(c_j, lca) \tag{3.8}$$

In this equation lca is the lowest common ancestor the categories have, so the distance is equal to the sum of the distances of the categories to that ancestor. In figure 3.1, 2 sample trees can be found for 2 main categories clothing and shoes. Distances between Chino and Legging is 2, with distances of 1 to the lca Pants. Chino and Pants is 1, Pants is the lca again here. Clothing and Shoes is 2 and Coats and Boots is 4, both with the root as the lca.

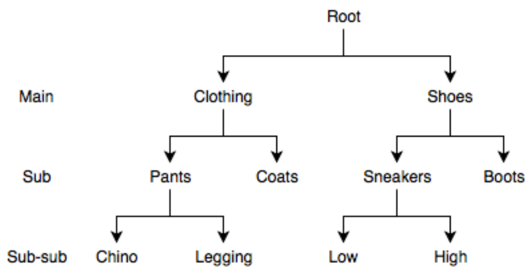


Figure 3.1: A sample category tree with 2 main categories Clothing and Shoes.

The distance between products p_1 and p_2 then becomes:

$$\begin{aligned} dist(p_i, p_j) = dist(c_i, c_j) + dist(b_i, b_j) + \\ dist(p_i, p_j) + dist(col_i, col_j) + dist(m_i, m_j) \end{aligned} \tag{3.9}$$

3.3.5. Clustering products for the recommendations

For different shopping patterns it was thought to be useful to have different approaches in recommendations. For example a user that is only looking at 1 type of product might benefit from other products of that type, while a user that has a more orienting style and is thus looking for multiple types of products might not benefit from recommendations of 1 type of product. We wanted to diversify the results for the latter user by means of clustering, while still showing useful products to the former user. This is achieved by determining the amount of centre products used for the clustering which conveniently matches with the user's interests given by their behaviour determined in section 3.3.1.

With the top 1-5 products from section 3.3.2 as centre products and a selection of other available products to use for the recommendations, clustering of the products happens with the similarity measure from section 3.3.4. They are then ordered by the distance, where again lower distance is better similarity. It can happen that 2 products have the same distance to the centre product and then a tie-break needs to happen. This can happen in 2 ways, either by price distance if used in a continuous manner or by popularity of the products if price is used as segments. Again, this will be determined later in section 5.4. The products that will be recommended to the user are selected by going over the 1-5 clusters and each time the highest product of the cluster will be selected until there are 5 products to recommend.

4 Behavioural recommender system at Fashionchick

In this chapter we will outline the platform where this research will take place. We talk about the use case of Fashionchick, how it differs from traditional webshops and what that means for their business model. After that, we will discuss the shop of the website itself and the recommender system that is used in the shop.

4.1. Use case: Fashionchick

Fashionchick.nl, owned by Sanoma Media Netherlands B.V., is a fashion platform targeted at women in the range of 20-50 years [1]. The website lists fashion and beauty products in the shop and offers fashion, styling and beauty advice by posting blogs, articles and advertorials according to the latest trends. Their goal is to inspire women to find the right items within the large number of fashion and beauty products that are available online today. Fashionchick has its own editors who write the pieces on the website and select products that go with it. These products are selected from the products listed in the shop on the website itself, giving the user a full experience within Fashionchick.nl. The products that are offered in the shop are not its own and no products are sold by Fashionchick. It is thus not a traditional webshop, but it lists products from other webshops and brands. It does this in a way the customer can quickly compare products from multiple sources. When a user would like to buy a product he or she can do so by clicking through to the corresponding webshop.

4.1.1. Business model

As Fashionchick does not sell products on the website the business model is different from a traditional e-commerce platform which does sell products. The source of income comes from the ad space and advertorials that are sold, but also from the traffic that is sent to the external webshops. Fashionchick closes a deal with a brand or a webshop, which then provides a feed of products to list on the website. When a user clicks on 1 of the products and is sent to the external webshop, the webshop pays the cost per click (CPC) of 32 cents to Fashionchick [2]. The compensation is agreed upon with over a hundred webshops. With feeds webshops provide, information that is displayed is kept up to date and matches the information on the original webshop.

This business model Fashionchick has, based on CPC, is a major difference with traditional webshops. For traditional webshops the conversion rate is measured by the number of purchases over the number of sessions, while for Fashionchick conversion rate is measured by the ratio of number of clickouts over the number of sessions. On Fashionchick.nl the customer journey is also different from a normal webshop. For a normal webshop this is a user browsing their products until a purchase is made and generally that is the end. For fashionchick it is different because a conversion, a click to an external webshop, is not the end of a journey. Users might return after they found out that the product followed was not what they were looking for after all and the journey continues.

4.1.2. Website

The website of fashionchick has 2 major parts, the inspiration section, which lists all the articles and advertorials, and the shop, which contains the products of the external webshops. Only the shop will be covered, as that is the relevant part for this research.

The shop is divided into fashion categories such as clothing, shoes or bags as well as accessories and beauty products. These categories all have a product listing page, which lists all the products for that category. An example for this can be seen in figure 4.1. On the left of this page, there is a selection box to narrow down the products with sub categories, as well as filter options beneath the sub categories, which are not visible in the screen capture. When a user hovers over a product tile, a button to go to the webshop of that product becomes visible. The user can either click that button or click on the rest of the tile to go to the product detail page within Fashionchick.nl.

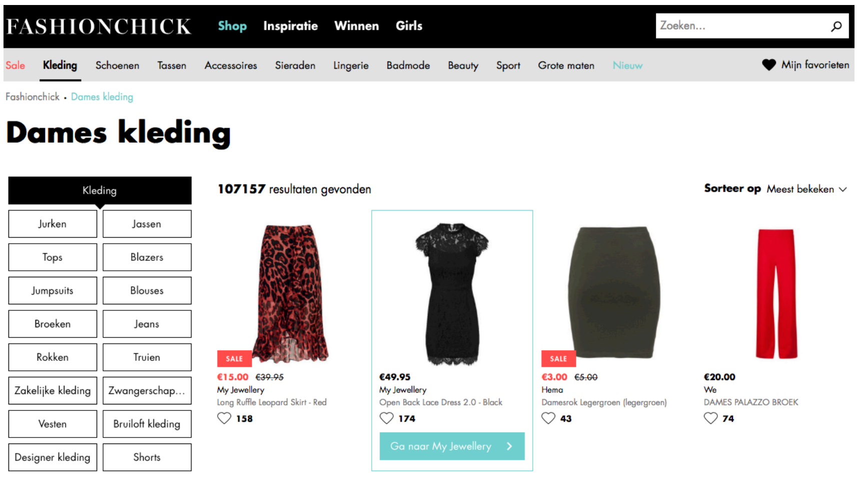
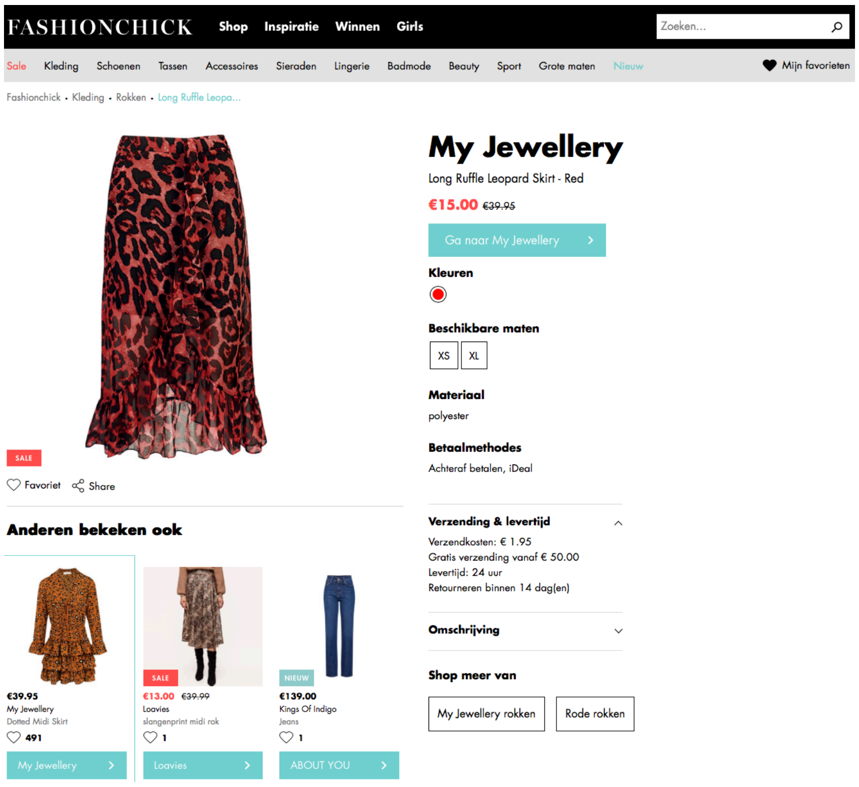


Figure 4.1: A category page for clothing.

On the product detail page, which can be seen in figure 4.2, extra information about the product can be seen, such as the current available sizes, attributes or shipping information. On a product detail page, there is a button to go to the webshop for that product. Because of the business model, Fashionchick always strives to increase the number of clicks that is sent to the webshops by making the website easy to navigate, so that people can find what they need easily.

The section called "Anderen bekeken ook", or "Others also viewed", on the product detail page is another way to try and help find interesting products. This section contains product recommendations based on the standard item-to-item collaborative filtering. On the desktop version, this box shows 3 products, whereas the mobile version shows a scrollable box which shows 5 products. As collaborative filtering is a method that relies on historical data about what products users look at together, these combinations might change overtime just as fashion trends change and thus products are combined with new, different products. Because this is updated regularly this means that there is data sparsity, which results in incomplete recommendations for the products. Not all products get recommendations, or products might get less than 3 or 5 products recommended.

Not showing products is something that is unwanted, as it prevents users to easily find other interesting products. To make sure there are always enough products to show on the product detail page, Fashionchick collects extra products to show when there are not enough products collected from the recommender system. It does this by looking at the most popular products from the lowest available sub category of the product that is being viewed. The recommendations box is then filled to 3 or 5 products with these most popular products. This ensures that there are always products shown, and these products are at



My Jewellery

Long Ruffle Leopard Skirt - Red

€15.00 ~~€39.95~~

Kleuren

Beschikbare maten

XS XL

Materiaal

polyester

Betaalmethodes

Achteraf betalen, iDeal

Verzending & levertijd

Verzendkosten: € 1.95
Gratis verzending vanaf € 50.00
Levertijd: 24 uur
Retourneren binnen 14 dag(en)

Shop meer van

Anderen bekeken ook

€39.95
My Jewellery
Dotted Midi Skirt
491

€13.00 ~~€39.99~~
Loavies
slangenprint midi rok
1

€139.00
Kings Of Indigo
Jeans
1

Figure 4.2: A product detail page.

least somewhat relevant to the user.

5 Offline data analysis

In this section we will talk about the offline data analysis that is necessary before the implementation for the live system can be done. We will analyse a dataset with click events to get a sense of user behaviour on Fashionchick.nl. We will also analyse what users view and which products they find more interesting. Lastly, we will analyse which price type to use for the product similarity measure from section 3.3.4.

5.1. Dataset

For the offline data analysis, we will get raw data which needs to be structured better. Sanoma has a storage of their own type of events called “SACEvents” with pageviews and other interactions with their sites. From this storage, a dataset of the first 2 weeks of September is collected. The choice for 2 weeks of data is made because of the A/B test that will be run in the end, which will also run for 2 weeks, and because of that gets 2 data points for each day. This is to eliminate chance in the results. This will be discussed more in section 7.1.4. The collected data is sorted on the timestamp. First it has to be split on the cookie identifier for each user and then it will be split again into separate browsing sessions. Splitting the sessions happens with a timeout set on 30 minutes, this is an often used approach [3, 23, 48]. The result then is a dataset of user sessions with a duration of at most 30 minutes.

The best possible ordering of the data is by timestamp, as it is the only way of telling what event happened before another. But consecutive pageviews might not be directly linked to each other. When we looked at the Google Analytics page of Fashionchick it became clear there is a significant part of the users that enter the website via google. So a plausible situation is that a user is searching on google, clicking on a link to Fashionchick, goes back to search results and then clicking on another link to Fashionchick. The result of this in the data is a sequence of pageviews like this: $S = \{p_1, p_2, p_3, p_4, p_5\}$. Where S is a session and p_n is a pageview. Now it might seem like these pages are linked in this order, but it is possible this navigational path is not possible via normal navigation on Fashionchick.nl. When a user visits sets of pages via google the data is actually more like this: $S = \{\{p_1, p_2\}, \{p_3, p_4, p_5\}\}$. Where the user entered on p_1 via google, clicked to another page, left Fashionchick and entered on p_3 via google again. Also, users can have multiple tabs open which makes the ordering even more complex and it is not possible to track how many tabs are open and when they are closed. Because of this the ordering and linking of pageviews will not be pursued any further and the assumption is made that the order is the same as the order obtained through sorting by timestamp. Because of the feature selection, as discussed in section 3.2.1, this will not be a problem.

The available fields that are important in the click event data for the analysis can be seen in table 5.1. There are more besides these, such as the information about A/B test groups and product information, but those will be discussed later when they are used.

There are a lot of events in the data that we do not need and will not use. Because of that we remove those before the analysis. We filter out all the events that have the value *rec_shown*

Field	Explanation
cookie	Unique identifier for a user on a single machine
event	Type of event. Options: pageview, clickout, rec_shown, rec_clicked
page_type	Type of the page. Options: frontpage, category, product, outpage, article, Question detail.
category	Full category string identifier with three levels, main, sub and sub sub categories. Different levels of categories are separated by commas.
scroll	The amount of horizontal and vertical scroll on a page.

Table 5.1: Explanation of important fields that are used.

or $rec\_clicked$ from the $event$ field as we do not do anything with the interaction with the recommendations yet. For the $page\_type$ column we remove the events where the value equals $frontpage$, $outpage$, $article$ or $question_detail$" because the focus for the behaviour is on the pages that are in the shop.

5.2. Detecting shopping behaviour

For the analysis of shopping behaviour we looked at the literature and found a great example in [49]. Based on that research we defined our own features to distinguish behaviour by in section 3.2.1, and with that information we can also define some expectations about what our expectations are relating to the first research question, **RQ1**: *How can browsing behaviour best be categorised and what shopping patterns can be discerned?* The aim of this research question is to find out how to group the behaviour of users into useful clusters, which can be used as a basis to recommend relevant products to those users. We expect to see similar clusters as in [49], where four notable clusters are found which are as follows: directed buying, search/deliberation, knowledge building and hedonic browsing. Patterns in those clusters are defined by the three major features, which can be seen in table 2.1. As our features are directly based on those features, we can compare our clusters to those shopping strategies and see if and how the feature patterns match. We will now move on to the clustering.

With the data split in separate sessions we start the analysis by extracting the features CV, PV and RPV from section 3.2.1 for each session. Table 5.1 already showed it: in the data from Fashionchick there are 3 levels of categories separated by commas. These are split for the analysis, and the feature CV is calculated for each level. These levels are treated as a tree structure, just as in section 3.3.4, where the main category is the parent of the sub categories and the sub-sub categories are children of a sub category. We then get features: CV_main, CV_sub and CV_sub_sub.

For the behaviour analysis, we do not want the category variety to become too strong, as there now are 3 features for that. To determine how to use these features, we looked at how the distribution of visits over the categories is for each level by calculating the entropy. Looking at the entropy of each category level gives us an idea whether or not we can focus on one of the features or if we lose information if we do that. For a user who looks at 5 different products, the sub-sub category can change while the sub and main categories stay the same. In reverse that is not possible, because when the main category changes the sub and sub-sub categories always change with it. Users can also visit the category listing page for a main category which results in empty entries for the sub and sub-sub categories. In table 5.2, we see the found entropy values and see, as expected, that the entropy for the sub-sub category is highest, main is the lowest and sub is in the middle. Because some users visit more product pages than others and there is quite a difference in entropy we do not want to use only 1 of these features, but we want to combine the variety features for the 3 levels by taking the average of the features CV_main, CV_sub and CV_sub_sub.

Category level	Entropy
Main	2.12
Sub	3.54
Sub-sub	4.42

Table 5.2: Entropy for the categories

Before we start the clustering of the sessions into groups based on behaviour, we want to see if there were major outliers by means of the variety features. Because the clustering uses the euclidean distance measure and CV and PV features are values between 0 and 1, the RPV feature needs to be normalised in order to not affect the results too much. Its outliers however negatively affect this because there might be some really large values leading to the majority of the sessions to fall into only a small portion of the normalised values. In the histogram in figure 5.1, we see the tail stretching to 30 with only a small frequency and there turned out to be 1 session with an RPV value of 350, which need to be filtered out. Judging from the histogram, a cap for RPV is set at 15. Sessions with an RPV value higher than 15 are removed.

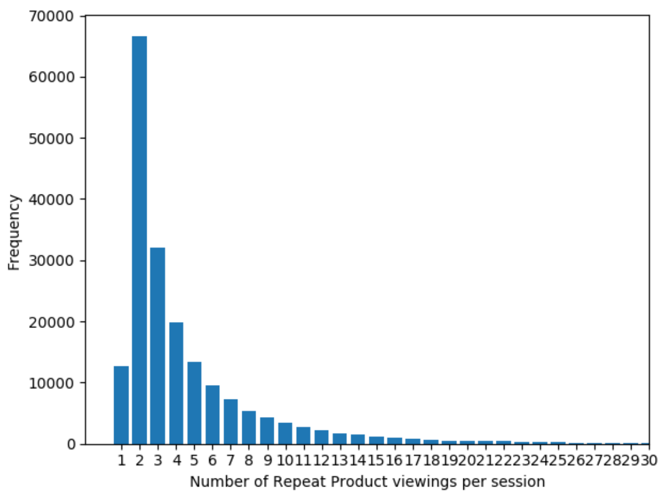


Figure 5.1: Repeat product viewings

We then start the k-means clustering. First, we have to set input values. The number of iterations is set at 100, as this was thought to be enough and the duration of the algorithm did not matter too much. The tolerance value, which makes sure the algorithm stops when there is not much change in the centre points, was set at 0.0001, as smaller values did not really result in a difference in clusters. Lastly, we do not really know yet what k, the number of clusters, should be so first we started with $k = 3$, then with 4 and with 5 to see how that would split the sessions into clusters. The results of this can be found in figures 5.2, 5.3 and 5.4.

These scatterplots show all of the sessions of the dataset on the 3 axes. The legends show the colours/shapes of the groups corresponding with the centre points and sizes of the groups. The scatterplots do not actually show the sessions split up into easily distinguishable separate clusters. As a matter of fact, when we look at plane like group of sessions on the low end of the product variety axis, the split runs with a straight line through these sessions. Because there are no distinguishable separate clusters this seems strange at first, but the split does make sense. In clusters used for behaviour classification we want to use as much of the variation in the features for classification, if then a group of sessions has the same value for one feature but varying values for the remaining two it is desirable to split on those

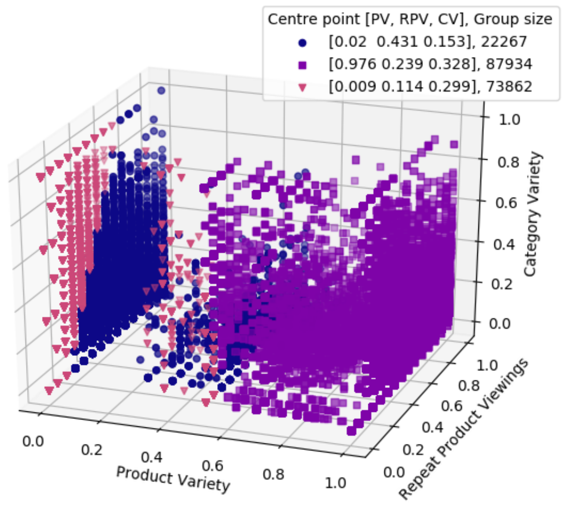


Figure 5.2: Clustering with k = 3.

features with great variation. In these scatterplots the sessions that have a value of zero for PV are spread out wide over the remaining two axes RPV and CV. A split through such a plane is what we want to see. One important thing to note is the plane of sessions for which the product variety is zero, or almost zero. This plane is the most disjunct of the rest because of the way the feature attributes the value zero when there are no product page views in a session, or a session just has a large value for RPV and a low value for PV. A gap then exists between these sessions and the other sessions, but still groups with a low product variety of both zero and non-zero values are made, which is good because this means the gap is not too big.

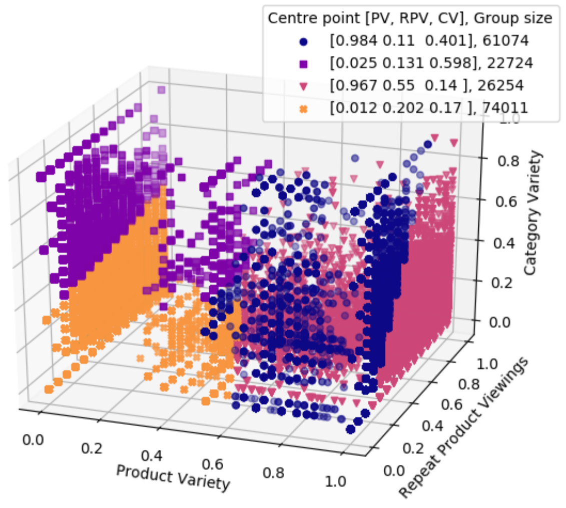


Figure 5.3: Clustering with k = 4.

Comparing the clusters between the k values 3, 4 and 5, it seems that 3 clusters is too little for a good separation. The reason that 3 is too little, is the large purple/square group in figure 5.2, which covers the whole axes of the RVP feature. Especially when the PV is high we want to distinguish users based on RPV because users that view a lot of different products can be separated by how many repeat viewings happen on a single product. This is an important indication of how directed a user is towards a single product or not. Then, moving on to 4 clusters in figure 5.3, we see that a split emerged in the plane with high PV which is what we want to see. However, looking at the centre points, we have 2 very similar groups, purple/square and orange/cross, which are almost only divided by the feature CV. In the scatterplot with 5 clusters in figure 5.4, these groups are further divided with a separate group which has low values for each of the 3 features. It seems only logical to have these sessions split off from the rest, as this is a group of sessions in which users only visited in a very shallow manner, which is a difficult group to recommend items for. Additionally, 5 clusters seems like a good split as the remaining 4 groups have a at most 1 of the 3 features being similar, where the other 2 features differentiate from each other. We did look for higher values of k, but that did not seem to result in more logical clusters of sessions with the centre points only becoming more similar. For that reason these plots are not included.

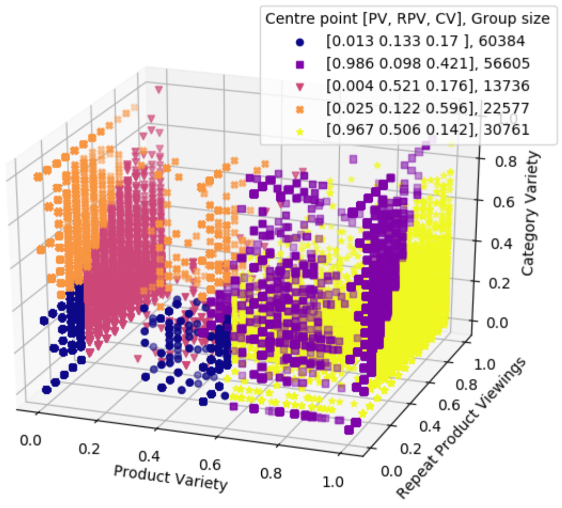


Figure 5.4: Clustering with k = 5.

We compare the centre points of the 5 clusters with the shopping strategies from [49] in table 2.1, where centre point values from 0 to 0.33 is low, 0.33 to 0.66 is moderate and 0.66 to 1 is high. We do see some similarities in behavioural patterns but not fully matching groups. In figure 5.4, with 5 clusters, we see a match of the yellow/star and blue/circle centre points with the search/deliberation and knowledge building patterns from table 2.1. The centre points of the other groups do not match a pattern of the shopping strategies in that same table. While there are around 4 groups, group number 1 is a group of shallow browsers, the feature values of some groups do match the behavioural patterns of the literature and some do not. It was not expected that they would perfectly match, as the measurements happened on different websites and in different domains. It is, however, interesting to see at least a somewhat similar gradation in how directed users are browsing the website. As a result of this, we cannot match the shopping strategies found in [49] in a 1-to-1 relation to the found clusters, and thus we will not use the same labels as in [49]. Despite that, we can identify users as browsing shallow, or looking at a high or low variety of products and categories. The 5 cluster centre points will be used in the online system to classify users into 1 of the groups. After this we will refer to the clusters with numbers in the order of appearance in figure 5.4.

5.3. User interests & product relevance

Next up is the method for finding products that are interesting for the user. As discussed in section 3.3.2, the features used to see whether a product is relevant to the user are: time, relevant time, scroll, number of pageviews for a product, the product sequence feature and the sequence combined with relevant time. These will all be scored with Pearson correlation against the conversion rate for products. As for Fashionchick, conversion rate is a bit different and is determined with the number of clickouts, and thus features will be correlated against the number of clickouts per product.

In table 5.3, the correlation scores can be found. We will now go over each score and talk

Feature	Correlation
npp	0.9468
tpp	0.1245
$rtpp$	0.4969
scr	0.9168
seq	0.7868
$rtseq$	0.7869

Table 5.3: Pearson correlation with the number of clickouts.

about what it actually means and if it is usable. First off we have the number of pageviews for a product. A correlation score of 0.94 is found, while it is really high, it does however seem artificial because clickouts can happen from category listings and product detail pages. So it is only natural that more views on the product detail page means more clickouts for that product. Next is the time on a product page. This is far lower than expected and thus not usable, not even with relevant time which is higher but not high enough. Then the amount of scroll on a page is also really high but also seems to be too directly linked to the number of pageviews. Then, lastly, we get to the product sequence features. These features are a bit more elaborate and that actually turned out pretty well. A correlation coefficient of 0.78 is a fairly strong positive correlation between both the features seq, $rtseq$ with the number of clickouts. Even though it is lower than some other features, this feature is less directly linked to the number of pageviews. In addition to that it can also differentiate between products better because of products coming earlier in the sequence getting a higher score. The difference between using and not using relevant time is minimal, but as there was a large difference between only using time and relevant time the feature product sequence with relevant time will be used to determine product relevance.

5.4. Product similarity

In section 3.3.4, it was discussed how the product similarity is determined, but the price feature is not definitive yet. There are 2 options to use the price, absolute difference and absolute difference between price segments. Both options have their advantages and disadvantages. To see which one we are going to use we want to see how each option matches to what users are looking at in a session. For that, we are going to look at the distributions of how much users look at products of a certain price and price ranges. Then, we are going to see what the median absolute distance is to the median for each session for both options [51]. This will show how close users stick to a certain price and what option fits better to select similar products.

The price ranges that will be used are the ones found on fashionchick.nl and each range is assigned an integer value and can be seen in table 5.4.

Range	Price values p
0	$p < 10$
1	$10 \leq p < 30$
2	$30 \leq p < 50$
3	$50 \leq p < 70$
4	$70 \leq p < 100$
5	$100 \leq p < 200$
6	$200 < p$

Table 5.4: Price ranges on fashionchick.nl.

Histograms with the distributions of the 2 options are made and these can be seen in figure 5.5. Only prices up to 200 euros are used for these histograms, that is why in the right histogram there is no 6th range.

In the histogram with the actual prices we see, as expected, an asymptotic distribution

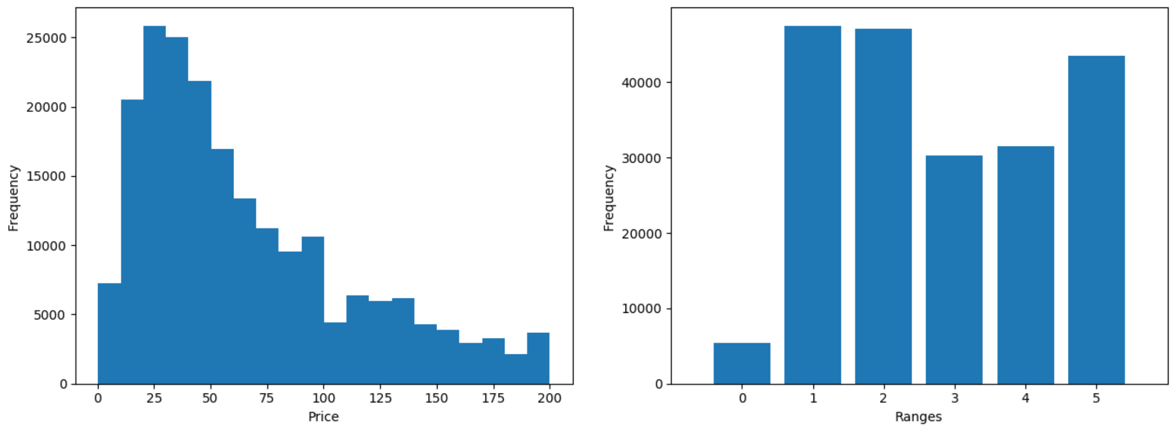


Figure 5.5: Price distributions of continuous values (left) and ranges (right).

where the majority of the products viewed are cheap and there are less products with higher prices. With the price ranges we see a much more evenly distributed histogram. That is most likely how the ranges were chosen to begin with. To see if this distribution of products over price ranges is more desirable, we calculate the median absolute deviation [51], we do this to the mean of prices and price ranges within each session and make histograms from the results. These histograms can be seen in figure 5.6.

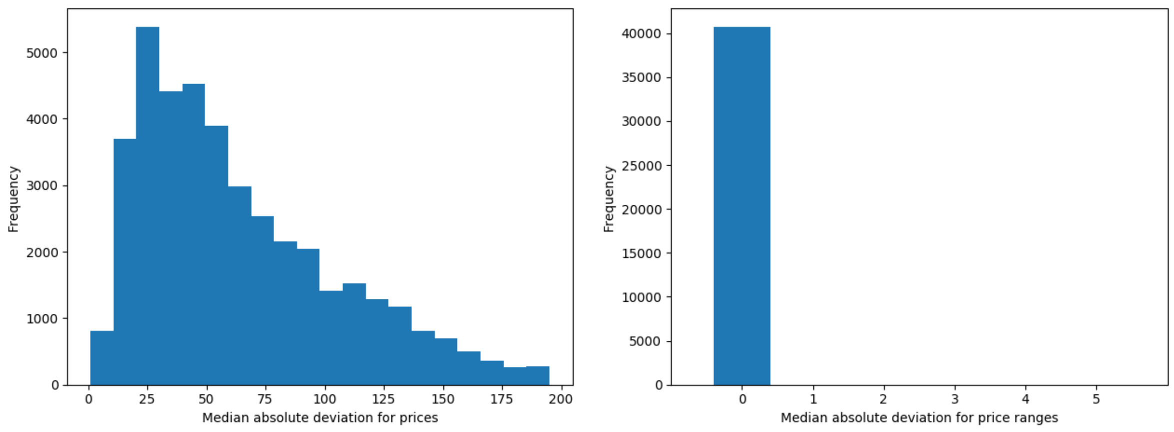


Figure 5.6: Distribution of median absolute distances to the median for continuous values (left) and ranges (right).

In the left figure we see for the absolute price distances that there is a large variety in median distances to the median, whereas for the price ranges this is equal to zero. This means that users do shop within the price ranges provided on the website and practically do not deviate. The conclusion can be drawn that this is a good measure to differentiate products which the user is or is not looking for. Therefore, we will use price ranges instead of the actual price as one of the distance features. Actual prices will also not be used as tie-breakers, instead, the popularity of a product will be used for this. Popularity is an entry in the data about a product in the ElasticSearch engine Fashionchick uses, more on that in chapter 6

6 Implementation live streaming & recommender system

After explaining the general approach for the behaviour driven recommender system in section 3.3, the main parts for the actual recommendations are clear. In this chapter the implementation of the live streaming and recommender system will be discussed. We will cover important decisions concerning the infrastructure and how to get the recommendations back to the website. First we will give a schematic overview of the full system with a short explanation, to get a better understanding of what components there are. After that we will go over each component in more detail and discuss the choices that were made.

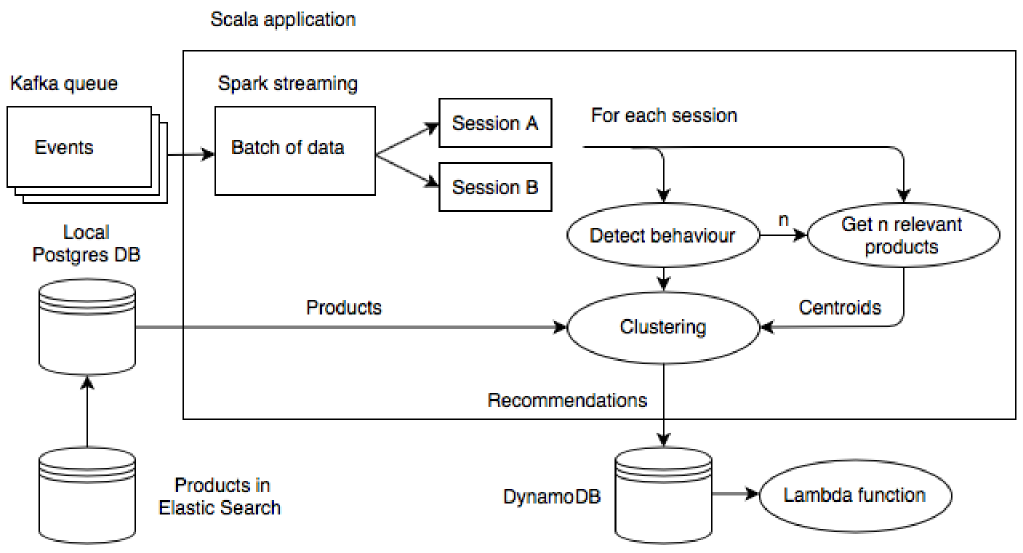


Figure 6.1: Schematic overview of the behaviour driven recommender system.

In 6.1 we see the schema of the implemented system. The great rectangle is the implemented application in Scala which starts a Spark stream [12]. All the components that are outside of this rectangle are external resources that the application communicates with. First, the connection from which the spark streaming gets its data is the Apacha Kafka queue, which was already present at Sanoma [7]. Then, each time a batch of data is collected, for each session that has new information the behaviour is detected and the clustering is started. For the clustering, products that are available on the Fashionchick website are used. These products are collected with a python script from Fashionchick's Elasticsearch engine [6] and stored in a local Postgres database. After the clustering, the top product for each cluster is selected and these products are stored in Amazon DynamoDB [5]. From here, the website can access these recommendations by making a call to an AWS Lambda function [8].

6.1. Spark streaming

For a system to make recommendations based on behavioural aspect within a user session and based on the products that a user views in that same session, a live streaming of events is necessary. Sanoma already had this running with an Apache Kafka platform. This queue contains SACEvents for all of Sanoma's websites, with information about the users, the website and a custom field which the website can fill in according to its needs. This queue can be live streamed by a Spark streaming application which we implemented in Scala. This application gets a batch of all the new data that entered the queue, since it registered to the queue, and this happens at a preset time interval. This interval can be set as big or as small as needed, but most important is that the collected batch is processed within the interval time. Spark streaming is a distributed data platform, but for this research only a single AWS instance was used, as that was enough to handle the computations in time. The interval time at which spark streaming collects the new batch of data that entered the queue is ideally set as low as possible to get data as often as possible. For development however, this interval was initially set at 30 seconds.

6.2. Feature extraction & user interest logging

With the streaming of the data in place we needed to filter the data stream to get only SACEvents for Fashionchick and extract the features as described in 3.2.1. Users are identified by a cookie stored on the user's machine, and sessions for these users are sets of events split by at most 30 minutes of inactivity, as discussed in section 3.2. For each session these features are extracted and stored in memory, as they are needed directly and every time new data for that session is contained in the new batch. To prevent the memory from getting full, we remove a session from memory and write its information to the disk when that session is timed out because it had no activity in the last 30 minutes.

Besides the feature extraction we also store information about the user's interests within the session. The SACEvents have information about which product pages the user visited. How relevant each product is, is determined by the product sequence with the relevant viewing time, as determined in section 5.3. Each product a user viewed then has a relevance score and the top n products with the highest score are selected to be centre points in the recommendation clustering, where n, the number of clusters, is set by the detected behaviour. With 1 of the centre points from figure 5.4 as the detected behaviour, n is equal to:

$$n = \max(\lfloor PV * 5 \rceil, 1)$$

Where PV is a feature from the detected behaviour and 5 is used because there are 5 products displayed on the product detail page. On the computer the website shows only 3 products, but 5 products are showed within a scrollable box on mobile. The maximum from the rounded value and 1 is taken because from the rounded value a zero can occur and we always need at least 1 product. It can happen that a user does not look at any product detail page and thus there are no products to use as centre points for clustering. In this case we take the most popular products of the top n categories a user looked at.

6.3. Behaviour detection

Each time a batch contains information about a session the new information is added to that session in memory and the features CV_main, CV_sub, CV_sub_sub, PV and RPV are updated. Because of the updated features the behaviour has to be detected again. This means that the detected behaviour is definitely subject to change as more information gradually comes available. Detecting the behaviour happens as described in section 3.3.1, and the cluster centre points from the clusters in figure 5.4 are used to do this. The 5 features CV_main, CV_sub, CV_sub_sub, PV and RPV are extracted rom the session information and the category features are combined into CV by taking the average again. The centre point to which the session has the lowest euclidean distance is stored within the session data for use during the recommendations process.

6.4. Recommendations clustering

Looking at the schema in figure 6.1, we still have to cover the lower part with the clustering. The clustering has 3 inputs: the detected behaviour, the centre point products and the products to be clustered. The detected behaviour influences how many relevant products are used as centre points and thus how many clusters there are. The top 5 visited categories are used to determine what products are selected from the local database for the clustering.

First, the local database is filled by taking a limited set of products from Elasticsearch. This search engine is also used by Fashionchick to list products on the website, but contains too many products to use in clustering. To limit the amount of products we first collect the 50 most popular products for each available sub-sub category, which are about 600 categories, and store these in the local Postgres database, which then contains approximately 30.000 products. Every 5 minutes, the script is run to collect new products and keep the database up to date, as products can sell out and become unavailable.

Which products are selected from the local Postgres database to use in the clustering depends on the focus of the session, which can either be on products or on categories. Clusters with a high RPV value were determined to have more focus on products and clusters with a low RPV value to have a higher focus on categories. The clusters and their focus can be seen in table 6.1. Note that the clustering is only started if the session has more than 1 page view and does not have the “shallow” focus for the detected behaviour. This decision was made because for these sessions there was not enough information to recommend items effectively.

Cluster	***PV***	***RPV***	***CV***	**Focus**	**Details**
1	0.013	0.133	0.169	Shallow	No particular direction.
2	0.986	0.098	0.421	Categories	Wide focus on multiple categories with multiple products.
3	0.004	0.521	0.176	Products	Very narrow focus on a single or some products, but multiple views for at least 1 product.
4	0.025	0.122	0.596	Categories	Wide focus and mainly viewing categories without clicking through.
5	0.967	0.506	0.142	Products	Narrow focus, multiple different products with repeat viewings.

Table 6.1: The 5 clusters and their focus.

For sessions with a focus on products, categories are selected by taking the categories from the lowest available category level from the n most relevant products. For sessions with a focus on categories, categories are selected by taking the 5 most viewed categories from a session. With the selected categories, queries are done to the local database to select the 50 most popular products for each category if it is a sub-sub or a sub level category and 100 if it is a main category. Depending on the session data, and how many products and categories are actually viewed, these can be less than 5 or n. The amount of selected products can thus vary from 50 to 500.

Then the selected products are clustered to the centre point products as described in section 3.3.5. When the clustering is done, for each cluster we take the product with the lowest distance to the centre point product and do this until we have 5 products to recommend to the user. Because there is no information about which product page the user will view next, and we do not want to recommend a product on its own product page, we also make a list of 5 backup products in the same way as the first 5 products. The backup list also contains 5 products as we want the product to be from the same product cluster as the product that is going to be replaced.

These 2 lists of products are stored in an AWS DynamoDB storage with the cookie identifier as a key. When a user then visits a product page the website can get the recommendations by calling the lambda function to access DynamoDB. The lambda function is called with the product id of the current product page, and with this the lambda function will check if that product is in the list of recommendations. When that happens, that product will be replaced with the corresponding product from the backup list.

7
Experimental setup

In this chapter, the experimental setup used to test the implemented system from chapter 6 will be discussed. With this test setup we will also try to find answers to the research questions. First, we will discuss the A/B testing platform used to divide users over separate variations, what type of recommendations will be served to each variation and what the duration of the test will be. We will also talk about how the test results will be logged and lastly we will cover the hypotheses of this research.

7.1. A/B testing

Sanoma has an online performance team which tests new features and visual adaptations on the websites before they go live and they make use of the VWO A/B testing platform [14]. With this platform we were able to tag users that visit a certain web page within the website. By giving them different tags, they are divided into groups and each group is served a different variation of the website. Users are tagged to be able to serve them the same variation when they visit the website later. For each variation, a percentage of the total user flow over that page can be set to manage how many users are tagged for each variation. This test is then run for a set amount of time to see wether or not there is a difference in interaction with the website between these groups. Interaction usually is the conversion rate, as that is what an e-commerce website would like to increase. VWO acts by running code on the web page to be tested and when that runs 2 things happen. First, the user is tagged, and then, based on the tag a user gets, the web page is altered. This all happens so quickly the user does not even know it happened. At Sanoma it was also setup to broadcast the tag into the SACEvents, so applications using the SACEvents know what variation a user is in.

7.1.1. Testing variations

On the product detail page of Fashionchick.nl there is the box with the recommendations through item-to-item collaborative filtering. We will test the behaviour driven recommendations in this location instead of the original recommendations Fashionchick currently uses. The main thing that needs to be tested in this A/B test is how well the behaviour driven recommender system performs compared to the item-to-item collaborative filtering. The first variation, will be the item-to-item collaborative filtering and will act as the Control group. The second variation, variation A, will be the behaviour driven recommender system. With the VWO platform a test is run on the product detail page and when a user visits that page he or she is tagged and an environment variable will be set so that the website knows which recommendations to collect. If the user is in variation A the website will make a call to the AWS Lambda function to get the behaviour driven recommendations generated by our system.

Besides the comparison between Control and variation A, we also want to test how well the selected recommendations method works for each detected behaviour group. We want to see if a wider variety in recommendations indeed matches the user behaviour to look at a wide variety of categories and vice versa. In order to test this we made a third variation, variation

B, in which the behaviour driven recommender system works the same as the one in variation A, except for the behaviour detection. In variation B we first detect the behaviour, and then randomly select one of the other behavioural clusters so that the recommendations method does not match the behaviour as originally detected. Recommendations are then generated based on the randomised behaviour cluster. If the user then opens a product detail page the recommendations are served to the user and the interaction is captured.

Implementation of variation B was a bit different because of how the VWO platform works. As VWO only tags users when they open a page that a test runs on, that is the first moment we can know in the backend of the streaming application what variation a user is in. In a test that requires randomisation of the assignment of the behaviour cluster and then generate recommendations based on that, knowing what variation a user is in the moment that user opens a product detail page is too late. For this the decision was made to run the test for variation B separate from the Control and A, and on the whole Fashionchick shop. A user then is tagged as soon as he or she opens any page of the shop and this makes sure there is more time before a user potentially opens a product detail page. Because this test was separate, extra care was needed for user tagging as the platform needed to be configured to only tag users in a way that they are mutually exclusive with the Control and variation A, and also that it has approximately the same amount of users. Each group was assigned to have a third of the users.

7.1.2. Hypotheses RQ2 and RQ3

Now that we know what variations we are going to test, we will define a set of hypotheses alongside the second and third defined research questions from section 1.2. We start with the second research question, **RQ2**: *How effective is recommending products based on the user's browsing behaviour?* For this question the hypotheses then become:

RQ2-H1: *Behaviour driven recommendations are more effective than traditional item-to-item collaborative filtering.*
RQ2-H2: *Recommending a narrowed selection of products is more effective for behaviour with a more directed focus.*
RQ2-H3: *Recommending a wider array of products is more effective for behaviour with a wider focus.*

This research question covers the recommendations of the system and more specifically the performance of selected recommendation methods to the corresponding behaviours. The first hypothesis should help determine how well the behaviour driven recommender system performs compared to the traditional approach. The approach in this research is experimental and it is useful to see how it compares to a well established recommender system. As we theorise that users visit the website with a certain intention, we expect behaviour driven recommendations to perform better than the traditional approach. Performance is measured by means of a set of metrics which indicate how the user interacts with the recommendations. More clicks on the recommended products means that these products make the user want to see more of it. We will talk more about the metrics that will be used in the next section, section 7.1.3. Hypotheses 2 and 3 focus the combination of the chosen methods with the detected behaviour and if these combinations are indeed appropriate. Other similar approaches to these kind of recommendations have proven to work [40, 42], and thus it is expected that the chosen methods work best with the corresponding behaviours.

Then, research question three, **RQ3**: *How feasible is a session based approach with real-time generation of recommendations?*

RQ3-H1: *The short time span of a browsing session is long enough to generate relevant recommendations to users individually.*

To test if a behaviour driven recommender system is feasible, a full working system was implemented. The last research question focusses on the temporal nature of the system. Making recommendations based on behaviour that happens live means working in a short time span and for all of the active users, which is a challenge. There was not much literature that did something similar as this research aims to do, so from a literature perspective it is hard to say wether it is feasible. However, with the chosen methods, which are very fast in computation time, we expect it to be feasible to exploit this short time span. This is reflected with the hypothesis for research question 3.

7.1.3. Testing metrics

To compare the three variations, we need metrics to measure how well each variation performs. As mentioned in the previous section, the conversion rate is usually the main feature on which the system is evaluated. In the end this is also true for this research, however, to be able to more deeply evaluate the performance of the system we define an extra set of metrics. As can be seen in table 5.1 in section 5.1, the event field has 2 values related to the recommendations, rec_shown and $rec\_clicked$. To understand these events we have to talk about how the recommender system on Fashionchick works. When a product detail page is loaded, it is checked if there are recommendations available for the current product. If they are available, they are loaded, if not, the most popular products within the lowest available category for the current product are loaded. Only when the recommender system actually had recommendations available the event rec_shown is fired, and $rec\_clicked$ when the user clicks on those recommendations.

Metric	**Details**
Total clickouts	The number of clickout events per session.
Product detail page clickouts	The number of clickout events from a product detail page per session.
Recommendation clickouts	The number of clickout events from a product detail page per session, where the clickout product id does not match that the product id of the pageview event before it.
Recommendations shown	The number of rec_shown events per session.
Recommendations clickthrough	The number of $rec\_clicked$ events per session.
Recommendations clickthrough ratio	The ratio of $\frac{rec\_clicked}{rec\_shown}$ events per session.
Clickouts after clickthrough	The number of clickout events from a product detail page per session, after a $rec\_clicked$ event occurred before it on a matching product id.

Table 7.1: Test metrics with a short explanation.

The metrics that will be collected and analysed can be seen in table 7.1. An important thing to note about the metrics is the difference between clickout and clickthrough: clickout events are actual SACevents that denote when a user clicks on a button to see the product on an external webshop, a clickthrough is a click on a product within the Fashionchick website. The clickout metrics are to see the difference between overall effect of the systems and more specific recommender related interactions, as clickouts can happen from category page product tiles and product detail pages, but also directly from the recommendation product tiles. The rec_shown and $rec\_clicked$ events related metrics are useful to analyse interaction with the recommendations within the website itself. A user clicking on the recommendations to view the product detail page within Fashionchick is also a positive interaction.

7.1.4. Test duration

For an A/B test it is important that there is enough data, which makes it possible to eliminate chance. We want to be as certain as possible that any differences between variations actually are because of the different types of recommendations. To achieve this the duration of the test is important. With the amount of visitors Fashionchick has in a week the test would have to run a little over a week in order to collect enough data, as calculated via [4]. Input values are omitted to protect business information of Fashionchick. The duration is rounded up to 2 weeks in order to get at least 2 data points for each day. This is important as websites typically have different visitor patterns on different days.

8 Results

In this section the results of the A/B test will be discussed. The test has run for 2 weeks, and in these 2 weeks the 2 variants have run beside the Control group while collecting the features described in section 7.1.3. In this section the features will be analysed by variant, then they will be split by type of behaviour to see how each type performed. Finally, they will be split by device because of the influence screen size has on how the website looks, which might impact the results.

8.1. Null hypothesis testing

To test wether or not the differences between the groups are significant, we apply the z-test for null hypothesis significance testing, as we are able to estimate the standard deviation for the groups. We calculate the z value for the z-test as follows [26]:

$$z = \frac{\hat{\mu_2} - \hat{\mu_1}}{\sqrt{\frac{\hat{\sigma_2}^2}{n_2} + \frac{\hat{\sigma_1}^2}{n_1}}} \tag{8.1}$$

In this equation $\hat{\mu}$ is the mean of the results for a group, $\hat{\sigma}$ is the standard deviation of the results for a group and n is the size of the group. With the z-test value, the significance p-value is retrieved with the survival function of the python package scipy.stats.norm [11]. For this test first a null hypothesis has to be set up, which is: "There is no significant difference in the number of clicks or clickouts between the 2 variations." The null hypothesis is rejected when the p-value is smaller than α which generally is set to one of 3 levels, 0.1, 0.05, 0.01. The difference in these levels represents a different amount of certainty that the found result is significant or not. In this thesis we will show the exact p-values in combination with a star (*) for each significance level. Significance values are not calculated for the rec_shown and $rec\_clicked$ metrics as it is not relevant to compare these on their own, because the different groups have different systems for generating recommendations influencing how often these events occur. Instead the fraction of $rec\_clicked$ over rec_shown is compared as this gives a better insight into what the ratio of the generated recommendations being clicked on is. We still show the average values per session for these metrics as it might be interesting to see how many recommendations are shown.

When the null hypothesis is rejected the difference between the groups is regarded as significant. However, with large sample sizes small differences are more quickly considered significant [58]. Because we run our test for 2 weeks, our sample sizes quickly become quite large, and for that we also look at the effect size of the differences. The effect size gives us insight into the practical significance [58]. Effect size, es, is calculated as follows [26]:

$$es = \frac{\hat{\mu_2} - \hat{\mu_1}}{\hat{\sigma}_p} \tag{8.2}$$

Where again, $\hat{\mu}$ is the mean for a group and here $\hat{\sigma}_p$ is the pooled standard deviation for the two groups. In our results we make a comparison between the Control group and variation A and a comparison between variation A and B. In equations 8.1 and 8.2 group 2 is the experimental group. In the 2 comparisons the experimental groups are A and B respectively. Seeing a positive effect size means that the experimental group outperforms the non-experimental group and vice versa for the negative effect.

8.2. Main results

For a total of approximately $60,000$ sessions that were in the A/B test that ran in the last 2 weeks of January, the metrics are collected and calculated per session. The mean over all sessions within each variation can be seen in table 8.1. To give some explanation about the values that are in the table and why some values are higher than 1 we turn to the conversion rate Fashionchick uses. For a normal webshop the conversion rate is related to a purchase and is usually between 0 and 1, because when a user makes a purchase that session ends. With the number of clickouts this value can exceed 1. This can happen because a user browses Fashionchick, clicks on a product to a webshop, does not like the item after all, returns to Fashionchick and then clicks out on another product, such a session has 2 clickouts.

	Control & Variation A Comparison				Variation A & B Comparison			
Metric	**Control**	**Var A**	**p-value**	**Effect Size**	**Var A**	**Var B**	**p-value**	**Effect Size**
Total clickouts	1.312	1.295	0.033^{**}	-0.007	1.295	1.178	0.000^{***}	-0.053
Product detail page clickouts	1.148	1.129	0.094^{*}	-0.009	1.129	1.031	0.000^{***}	-0.05
Recommendation clickouts	0.026	0.024	0.248	NA	0.024	0.019	0.002^{***}	-0.019
Recommendations shown	1.310	1.234	NA	NA	1.234	1.151	NA	NA
Recommendations clickthrough	0.089	0.063	NA	NA	0.063	0.048	NA	NA
Recommendations clickthrough ratio	0.042	0.039	0.000^{***}	-0.068	0.039	0.030	0.000^{***}	-0.045
Clickouts after clickthrough	0.171	0.162	0.002^{***}	-0.019	0.162	0.134	0.000^{***}	-0.066

Table 8.1: Main results of the A/B test. Two comparisons are made, between Control and variation A, and between variation A and B. Note that the two variation A columns contain the same values. P-values are shown with a star rating for the three levels of α. Effect size is only calculated when there is at least one star.

In the overall results in table 8.1 we get a general idea of performance of the 3 variations. Looking at the metrics calculated for the Control group we see that some metrics are quite low. For example, for clickouts directly from the recommendations, this means that it only happens for approximately 2% of the sessions. Looking at the comparison between Control and variation A, variation A seems to be quite close to the Control group. Even though it seems the null hypothesis can be rejected, for most metrics the effect sizes are low. When we look at the comparison between variations A and B the differences seem to be somewhat bigger. This is something reflected slightly by the effect sizes while they are still quite low. In clickouts from the product detail page this difference is the most apparent. Overall the differences in effect of the behaviour driven recommender system with the traditional collaborative filtering approach seem quite slim. To better analyse the results we will split the data by device the user uses and different detected behaviour types.

8.3. Results split by device

For the result data split by device there are three types of devices that can be detected: mobile, tablet and computer. This split is relevant as the different device types have greatly varying screen sizes which influence how the website is displayed. On a product detail page on mobile for example the recommendations are not even visible when opening the page and can only be seen after scrolling. The results for tablet will not be shown here, as the null hypothesis for comparisons between variations on tablet are seldom rejected. The biggest differences were found for the sessions that happened on the computer which can be seen in table 8.2.

If we look at the comparison between Control and variation A for sessions on the computer, we see that variation A performs worse than the Control group. It scores lower on all metrics and according to the p-values the differences are significant. The effect sizes however, are still

	Control & Variation A Comparison				Variation A & B Comparison			
Metric	**Control**	**Var A**	**p-value**	**Effect Size**	**Var A**	**Var B**	**p-value**	**Effect Size**
Total clickouts	1.430	1.298	0.000^{***}	-0.06	1.298	1.160	0.000^{***}	-0.077
Product detail page clickouts	1.328	1.182	0.000^{***}	-0.073	1.182	1.087	0.000^{***}	-0.058
Recommendation clickouts	0.084	0.053	0.001^{***}	-0.051	0.053	0.045	0.103	NA
Recommendations shown	1.469	1.405	NA	NA	1.405	1.262	NA	NA
Recommendations clickthrough	0.146	0.069	NA	NA	0.069	0.043	NA	NA
Recommendations clickthrough ratio	0.060	0.045	0.000^{***}	-0.159	0.045	0.029	0.000^{***}	-0.093
Clickouts after clickthrough	0.195	0.122	0.000^{***}	-0.148	0.122	0.125	0.338	NA

Table 8.2: Results for sessions on computer.

low, the clickthrough ratio for the recommendations and the clickouts after this clickthrough are the only ones to exceed 0.1, but those features had really low values to begin with. This means that there might not be much to gain for those features. The low effect sizes are most easily explained by the high standard deviations that were found for result samples.

Looking at the comparison between variation A and B it seems that overall variation B scores less than variation A. This is most evident for the product detail page clickouts and the recommendations clickthrough ratio. For the differences between other metrics related to recommendation (thus not looking at the total number of clickouts) the null hypothesis cannot be rejected and because of that, there seem to be no significant differences.

8.4. Results split by behaviour

Lastly, we split up the results by type of behaviour. For this, we will only look at the comparison between variation A and B and leave the Control group out. These comparisons are useful to determine how well the chosen approaches perform for the corresponding behaviour groups. The general idea is that a user who is looking at a wide variety of products or categories also benefits from a wider offer of products in the recommendations, and a user who is directed at a certain type of product benefits from an offer of other similar products. This is thus tested by mixing up the chosen approach with the randomised behaviour in variation B. The splits by behaviour happen on actually detected behaviour instead of randomised behaviour types of variation B, because variation A does not have these randomised behaviour types. Division over the different behaviour clusters can be seen in table 8.3:

Detected behaviour cluster	Percentage of users in cluster
2	$\approx 40\%$
3	$\approx 0.5\%$
4	$\approx 2\%$
5	$\approx 46\%$

Table 8.3: Division over the behaviour clusters. The clusters 2, 3, 4 and 5 are the same clusters as in 5.4. Cluster 1 is not shown as no recommendations are generated for that cluster.

What immediately becomes clear is that the amount of sessions per cluster is quite different from the clustering from figure 5.4. Clusters 3 and 4 have a significantly smaller proportion of the sessions. This is likely explained by a time gap between the clustering based on offline data and the A/B test. In hindsight the difference between behaviour is much larger than expected. Why this time gap has such a big influence on how sessions are classified can possibly be explained by the short nature of a session. With sessions having a length of only approximately 5-15 pages, classification can shift fairly quickly. In the future clusters used for a system like this should be generated as close to the live test as possible to eliminate these effects. In addition to that another approach to managing session data could be taken. More on these ideas in section 9.2. Because of the small number of sessions we will not discuss the results of sessions classified in behaviour clusters 3 and 4 and we will still look at clusters 2 and 5.

The results in table 8.4 show a significantly lower amount of clickouts per session than other splits. This is explained by the length of the sessions in this split, which averaged

Metric	Var A	Var B	P-Value	Effect Size
	Variation A & B Comparison			
Total clickouts	0.802	0.740	0.000^{***}	-0.038
Product detail page clickouts	0.662	0.621	0.014^{**}	-0.027
Recommendation clickouts	0.013	0.008	0.001^{***}	-0.039
Recommendations shown	0.950	0.848	NA	NA
Recommendations clickthrough	0.053	0.038	NA	NA
Recommendations clickthrough ratio	0.038	0.028	0.000^{***}	-0.059
Clickouts after clickthrough	0.057	0.046	0.000^{***}	-0.048

Table 8.4: Results for sessions classified in cluster 2.

around 5 pageviews. This average was around 10 pageviews for the sessions classified in cluster 5, and we also see this in the amount of clickouts per session for those sessions, which is twice as high. So while the average per session is lower for sessions in this cluster the difference between variation A and B is visible and deemed to be significant. However, the effect sizes for the differences between these groups seem to imply the differences are minor.

If we then look at the results for sessions classified in cluster 5 in table 8.5, we continue to see marginal differences between variation A and B. Where variation A performs slightly better than variation B, we see the biggest differences in the product detail page clickouts and clickouts from the detail page after a clickthrough on the recommendations, and even here the effect sizes are low.

Metric	Var A	Var B	P-Value	Effect Size
	Variation A & B Comparison			
Total clickouts	1.609	1.496	0.000^{***}	-0.05
Product detail page clickouts	1.418	1.316	0.000^{***}	-0.054
Recommendation clickouts	0.036	0.030	0.045^{**}	-0.015
Recommendations shown	1.441	1.417	NA	NA
Recommendations clickthrough	0.065	0.054	NA	NA
Recommendations clickthrough ratio	0.039	0.030	0.002^{***}	-0.035
Clickouts after clickthrough	0.204	0.179	0.000^{***}	-0.052

Table 8.5: Results for sessions classified in cluster 5.

9 Conclusion & future work

In this chapter we will return to the research questions and answer them. With that, we conclude this thesis. First we will answer the three research questions and after that we will answer the main research question of this thesis. Then we will discuss areas of interest we think have potential for future work.

9.1. Conclusions

In this thesis we have implemented and tested a fully working behaviour driven recommender system in the fashion domain at Fashionchick.nl. The first step was to research user behaviour on the website. We have mapped the behaviour of users into useful information for this recommender system. We did this by clustering user sessions with the k-means clustering algorithm on the variety of product pages, category pages and the number of repeat product viewings. We already discussed the clustering in section 5.2, but we will now answer the first research question based on the clustering. The first research question, **RQ1**, is:

RQ1: *How can browsing behaviour best be categorised and what shopping patterns can be discerned?*

Categorising browsing behaviour by means of clustering on the three selected features gave pretty interesting results. Though there were no apparent clusters in the scatterplot of figure 5.4, the splits in the grouped sessions and the centre points of the clusters did show quite varying patterns. When we compared the found patterns to the patterns found in the literature we did not find fully 1-to-1 matches and thus decided not to use the same labels as were used in [49]. There were however similar patterns and the best matches were with search/deliberation and knowledge building. With the 5 found clusters we can conclude that browsing behaviour can be categorised with the k-means clustering in a very varied manner. While the patterns did not fully match our expectations based on the literature, they do represent varying levels of direction of the user to products and categories. With that we found an answer to our first research question.

Then, moving on to the second research question, **RQ2**, with corresponding hypotheses:

RQ2: *How effective is recommending products based on the user's browsing behaviour?*

RQ2-H1: *Behaviour driven recommendations are more effective than traditional item-to-item collaborative filtering.*
RQ2-H2: *Recommending a narrowed selection of products is more effective for behaviour with a more directed focus.*
RQ2-H3: *Recommending a wider array of products is more effective for behaviour with a wider focus.*

This research question was addressed in the A/B test in section 7.1. In this A/B test we looked at the performance of our implemented behaviour driven recommender system compared to a traditional collaborative filtering approach. The differences between the Control group and variation A were overall quite slim, but were consistent. Because variation A scored lower on most features, and these differences were found to be significant, we have to conclude that our system performs slightly worse than the collaborative filtering. With this information we have to reject our hypothesis **RQ2-H1**. The A/B test also tested how well the chosen recommendation approach fit the behaviour patterns, it did that with the addition of variation B. Though the results did not cover all behaviour clusters, the two clusters, 2 and 5, are quite different in the level of direction. Where cluster 5 is more directed and cluster 2 has a wider focus. In both groups the differences between variation A and B again were quite slim, considering the effect sizes, but variation A did consistently outperform variation B. The differences between the two variations were significant according to the p-values. This suggests that the chosen recommendation methods do fit the behaviour patterns and that both hypotheses **RQ2-H2** and **RQ2-H3** should not be rejected. This proves that while this system may not be ready for production yet, it does have potential. With that we have answered our second research question.

Then we address our third research question, **RQ3**, with corresponding hypothesis:

RQ3: *How feasible is a session based approach with real-time generation of recommendations?*

RQ3-H1: *The short time span of a browsing session is long enough to generate relevant recommendations to users individually.*

The A/B test ran for two weeks and in that time handled around 60,000 sessions. For each session the system collected the data, detected the behaviour and generated recommendations. All this happened multiple times per session and considering the results, the recommendations were at least relevant to some extent. While the system is able to handle the amount of sessions on Fashionchick.nl and is able to generate recommendations in time for a large enough part of the sessions to lead to the found results, we still reject the hypothesis **RQ3-H1**. There are ways to improve the relevance of the recommendations and we think this is necessary for a system as this one to be used in production. We will go into more detail in the next section, section 9.2, but we think a lot of the improvement can be achieved by collecting data about earlier sessions of a user.

Lastly, we turn to our main research question, and we will answer it based on our answers to the other research questions:

Main Research Question: *How well does a behaviour driven recommender system perform as an alternative to traditional item-based collaborative filtering in the fashion domain?*

The main part of this research question, the comparison between our system and the traditional item-based collaborative filtering, is already addressed with hypothesis **RQ2-H1**, which we had to reject. The results pointed out that our system did not perform as well as the item-based collaborative filtering, but the differences were quite small. This means that the system does have potential and at the same time needs more work.

Detecting browsing behaviour and using that in combination with session data as input for a recommender system is a method which, in our opinion, should be pursued further. The found clusters were very diverse and gave different angles to use for recommendations. However, we think the approach with information about just a single session does not provide enough information to consistently generate relevant recommendations.

To conclude, we can say that the implemented behaviour driven recommender system is not ready yet to be deployed as an alternative to item-based collaborative filtering in the fashion domain. It performed almost as good as the collaborative filtering, the differences were quite small. If we look at the additional test of the A/B test we can say that the system does have a positive effect and that it is a good starting point to be improved upon.

9.2. Future work

We successfully implemented a fully working behaviour driven recommender system. We think this direction is promising and in this section, we talk about some area's in which we think the system could be improved.

The first area where we would like to improve the system is the timeframe the recommender system works in. This thesis proved that a session based approach can work. However, it is quite limiting in a couple of ways. First of all, the time available to generate recommendations is quite short, therefore it is possible that some generated recommendations were not finished in time before the user opened a page which could display them. Secondly, the information that is available to the system can be very limited. Our approach only uses information about the current session, which can decrease the relevance of the results. For example, because of the sparsity of the available data the detected behaviour, as well as interesting products, can change every pageview, possibly leading to unreliable recommendations. The way we would extend the timeframe and with that improve the system, is by using multiple user sessions stored in a Markov Model. With this, the system would be able to use previous information about user behaviour and product interests to recommend items in the early part of the current session, decreasing the sparsity of the available data to the system. In addition to that, the previous behaviour of users in the Markov Model can be used to predict future behaviour of users. With that prediction the system does not have to wait until there is enough information about the current session before generating the first recommendations and can even generate the first recommendation just after the previous session has ended.

Another area in which we would improve the system is in the method for recommendations. In this research, our approach was to use only information of the current user, as we wanted base the recommendations on the behaviour of the user only. Because of that we chose not to use collaborative filtering, as that uses information of other users as well, which might not be applicable to the current intentions of the user. Now that we have built and tested this prototype and have shown that it works to some extent, we think there are ways to combine it with collaborative filtering. With the classification of users into behaviour groups, collaborative filtering could be applied within these groups. This would make sure the general range of the search matches between users. In addition to that, collaborative filtering could also be combined with the first improvement. Because it might be a good approach to recommend items relevant for the predicted behaviour, when a user did not look at any items yet. These recommendations can then be based on similar users within that same behaviour group.

Our current approach to recommending products based on user behaviour was to divide the users into behaviour groups. This allowed us to define a limited number of general approaches of recommending products based on each type of behaviour. This also allowed a more simple comparison of the effect of the system between behaviour types. However, this generalisation makes the recommendations fit less to a specific user. Users might benefit more from an even more personal approach, by leaving out the detection of the behaviour. Instead of using the centre points of the detected cluster as input information for the clustering, the collected features for a user session can be used directly. This would also ask for a more robust recommender system which can generate recommendations in all edge cases of user behaviour, but would make for more fitting recommendations for each user individually.

The features used for product similarity were used in a rather simple way. For most features we chose to simply match them and give them a score for a full match or not. However, these could also be used differently to get more accurate similar products. The first feature we could have done different is the price. The price can be used absolute and in price ranges. We chose to use it with price ranges because people look at products roughly within the same price range. Distance between 2 prices can also be measured absolute, which would give more varying distances between products. Another feature we could approach differently is the colour. Now colour is also scored as a direct match or not, but that does not fit the feature very well. Colours distances between black and blue would be the same as blue and orange. Instead we could take RGB values for colours and calculate euclidean distances between those values. This would make for much more accurate similarities between products based on colour.

A
Other A/B test results

In this chapter we will list the tables with results of data splits that were not included in chapter 8. In these results there are either no significant differences to show and evaluate or there were too little data points left in a split.

A.1. Results split by device

	Control & Variation A Comparison				Variation A & B Comparison			
Metric	**Control**	**Var A**	**p-value**	**Effect Size**	**Var A**	**Var B**	**p-value**	**Effect Size**
Total clickouts	1.309	1.313	0.353	NA	1.313	1.204	0.000^{***}	−0.047
Product detail page clickouts	1.133	1.136	0.435	NA	1.136	1.037	0.000^{***}	−0.048
Recommendation clickouts	0.016	0.021	0.002^{***}	0.022	0.021	0.014	0.001^{***}	−0.023
Recommendations shown	1.301	1.221	NA	NA	1.221	1.153	NA	NA
Recommendations clickthrough	0.078	0.061	NA	NA	0.061	0.047	NA	NA
Recommendations clickthrough ratio	0.038	0.037	0.000^{***}	−0.048	0.037	0.029	0.000^{***}	−0.042
Clickouts after clickthrough	0.172	0.173	0.388	NA	0.173	0.140	0.000^{***}	−0.076

Table A.1: Results for sessions on mobile.

	Control & Variation A Comparison				Variation A & B Comparison			
Metric	**Control**	**Var A**	**p-value**	**Effect Size**	**Var A**	**Var B**	**p-value**	**Effect Size**
Total clickouts	1.025	1.009	0.264	NA	1.009	0.913	0.000^{***}	−0.063
Product detail page clickouts	0.862	0.877	0.356	NA	0.877	0.827	0.085^{*}	−0.039
Recommendation clickouts	0.002	0.002	0.472	NA	0.002	0.008	0.005^{***}	0.065
Recommendations shown	1.004	0.964	NA	NA	0.964	0.850	NA	NA
Recommendations clickthrough	0.094	0.074	NA	NA	0.074	0.075	NA	NA
Recommendations clickthrough ratio	0.057	0.050	0.113	NA	0.050	0.049	0.473	NA
Clickouts after clickthrough	0.086	0.097	0.152	NA	0.097	0.088	0.212	NA

Table A.2: Results for sessions on tablet.

A.2. Results split by behaviour type

	Variation A & B Comparison			
Metric	**Var A**	**Var B**	**P-Value**	**Effect Size**
Total clickouts	5.309	2.961	0.000^{***}	-0.279
Product detail page clickouts	4.742	2.750	0.000^{***}	-0.28
Recommendation clickouts	0.010	0.039	0.015^{**}	0.159
Recommendations shown	3.330	2.684	NA	NA
Recommendations clickthrough	0.381	0.250	NA	NA
Recommendations clickthrough ratio	0.044	0.046	0.152	NA
Clickouts after clickthrough	1.175	0.776	0.000^{***}	-0.353

Table A.3: Results for sessions classified in cluster 3.

	Variation A & B Comparison			
Metric	**Var A**	**Var B**	**P-Value**	**Effect Size**
Total clickouts	1.911	1.730	0.063^{*}	-0.048
Product detail page clickouts	1.691	1.539	0.158	NA
Recommendation clickouts	0.003	0.008	0.042^{**}	0.077
Recommendations shown	1.481	1.385	NA	NA
Recommendations clickthrough	0.127	0.075	NA	NA
Recommendations clickthrough ratio	0.059	0.028	0.027^{**}	-0.115
Clickouts after clickthrough	0.306	0.302	0.427	NA

Table A.4: Results for sessions classified in cluster 4.

Bibliography

[1] Fashionchick over ons. https://www.fashionchick.nl/over-ons.html, . Accessed: 2018-03-05.

[2] Fashionchick verdienmodel. https://www.fashionchick.nl/service/verdienmodel-fashionchick, . Accessed: 2019-03-06.

[3] How a web session is defined in analytics. https://support.google.com/analytics/answer/2731565?hl=en. Accessed: 2019-04-19.

[4] A/b test size calculator. https://abtestguide.com/abtestsize/. Accessed: 2019-04-23.

[5] Amazon dynamodb. https://aws.amazon.com/dynamodb/. Accessed: 2019-04-17.

[6] Elasticsearch: Restful, distributed search & analytics. https://www.elastic.co/products/elasticsearch. Accessed: 2019-04-17.

[7] Apacha kafka, a distributed streaming platform. https://kafka.apache.org. Accessed: 2019-04-17.

[8] Aws lambda - serveless compute. https://aws.amazon.com/lambda/. Accessed: 2019-04-17.

[9] Netflix main website. https://www.netflix.com/browse, . Accessed: 2019-02-06.

[10] Netflix creates 1 million dollar netflix prize to promote progress in recommendation systems. https://media.netflix.com/en/press-releases/netflix-creates-dollar1-million-netflix-prize-to-promote-progress-in-recommendation-systems-migration-1, . Accessed: 2019-02-01.

[11] scipy.stats.t - scipy v1.3.0 reference guide. https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.norm.html. Accessed: 2019-05-28.

[12] Spark streaming | apache spark. https://spark.apache.org/streaming/. Accessed: 2019-04-22.

[13] Growth statistics of the dutch online shopping market in 2017. thuiswinkel.org. https://www.thuiswinkel.org/nieuws/3710/nederlandse-consumenten-besteedden-22-5-miljard-online-in-2017. Accessed: 2018-03-12.

[14] All-in-one a/b testing and conversion optimization platform™ | vwo. https://kafka.apache.org. Accessed: 2019-04-17.

[15] 300 hours of video is uploaded to youtube every minute. https://tubularinsights.com/youtube-300-hours/. Accessed: 2019-02-01.

[16] G Adomavicius and a Tuzhilin. Toward the Next Generation of Recommender Systems: a Survey of the State of the Art and Possible Extensions. *IEEE Transactions on Knowledge and Data Engineering*, 17(6):734–749, 2005. ISSN 10414347. doi: 10.1109/TKDE.2005.99. URL http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1423975.

[17] Eugene Agichtein, Eric Brill, and Susan Dumais. Improving Web Search Ranking by Incorporating User Behavior Information General Terms. Technical report, 2006. URL http://susandumais.com/sigir2006-fp345-ranking-agichtein.pdf.

[18] Sagar Arora and Deepak Warrier. Decoding Fashion Contexts Using Word Embeddings. *KDD Fashion Workshop*, 2016. URL https://kddfashion2016.mybluemix.net/kddfashion{_}finalSubmissions/DecodingFashionContextsUsingWordEmbeddings.pdf.

[19] Ricardo Baeza-Yates, Liliana Calderón-Benavides, and Cristina González-Caro. The Intention Behind Web Queries. *String Processing and Information Retrieval*, 4209:98–109, 2006. ISSN 03029743. doi: 10.1007/11880561_9. URL http://www.springerlink.com/content/y1287823117n33p8/.

[20] John S Breese, David Heckerman, and Carl Kadie. Empirical Analysis of Predictive Algorithms for Collaborative Filtering. Technical report, 1998. URL https://arxiv.org/pdf/1301.7363.pdf.

[21] Andrei Broder. A taxonomy of web search. *ACM SIGIR Forum*, 36(2):3, 2002. ISSN 01635840. doi: 10.1145/792550.792552. URL http://portal.acm.org/citation.cfm?doid=792550.792552.

[22] Michael Buckland and Fredric Gey. The Relationship between Recall and Precision. Technical report, 1994. URL https://onlinelibrary.wiley.com/doi/pdf/10.1002/(SICI)1097-4571(199401)45:1{%}3C12::AID-ASI2{%}3E3.0.CO;2-L.

[23] Jia Carpenter, William J and Ji, Hai and Ji, Zi Jian and Li, Yuan Yuan and Ma, Wen Bo and Mi. Cookie based session timeout detection and management., 2018. URL https://patentimages.storage.googleapis.com/b8/21/82/cbb35220dc6f3a/US20180309836A1.pdf.

[24] Lara D Catledge and James Edward Pitkow. Characterizing browsing behaviors on the world-wide web. Technical report, Georgia Institute of Technology, 1995.

[25] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation. Technical report, 2014. URL https://arxiv.org/pdf/1406.1078.pdf.

[26] Robert Coe. It's the Effect Size, Stupid. What effect size is and why it is important. Technical report, 2002. URL http://www.cem.org/attachments/ebe/esguide.pdf.

[27] Robert Cooley, Bamshad Mobasher, and J Aideep Srivastava. Data Preparation for Mining World Wide Web Browsing Patterns. Technical report, 1999. URL https://link.springer.com/content/pdf/10.1007/bf03325089.pdf.

[28] Jesse Davis and Mark Goadrich. The Relationship Between Precision-Recall and ROC Curves. 2006.

[29] Joaquin Delgado and Ishii. Memory-Based Weighted-Majority Prediction for Recommender Systems. Technical report, 1999. URL https://pdfs.semanticscholar.org/de02/9d066fa3114252a1d6e2b11e1b654d55fa7c.pdf.

[30] Lucky Dhakad, Mrinal Das, Chiranjib Bhattacharyya, Samik Datta, Mihir Kale, and Vivek Mehta. SOPER: Discovering the Influence of Fashion and the Many Faces of User from Session Logs using Stick Breaking Process. *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management - CIKM '17*, pages 1609–1618, 2017. doi: 10.1145/3132847.3133007. URL http://dl.acm.org/citation.cfm?doid=3132847.3133007.

[31] Amy Wenxuan Ding, Shibo Li, and Patrali Chatterjee. Learning user real-time intent for optimal dynamic web page transformation. *Information Systems Research*, 26(2):339–359, 2015. ISSN 15265536. doi: 10.1287/isre.2015.0568. URL http://pubsonline.informs.org.https://doi.org/10.1287/isre.2015.0568.

[32] Antonino Freno. Practical Lessons from Developing a Large-Scale Recommender System at Zalando. *Proceedings of the Eleventh ACM Conference on Recommender Systems - RecSys '17*, pages 251–259, 2017. doi: 10.1145/3109859.3109897. URL http://dl.acm.org/citation.cfm?doid=3109859.3109897.

[33] Gene Golub and William Kahan. Calculating the Singular Values and Pseudo-Inverse of a Matrix. Technical Report 2, 1965. URL http://www.siam.org/journals/ojsa.php.

[34] Carlos A. Gomez-Uribe and Neil Hunt. The Netflix Recommender System. *ACM Transactions on Management Information Systems*, 6(4):1–19, 2015. ISSN 2158656X. doi: 10.1145/2843948. URL http://dl.acm.org/citation.cfm?doid=2869770.2843948.

[35] J A Hartigan and M A Wong. Algorithm AS 136: A K-Means Clustering Algorithm. Technical Report 1, 1979. URL https://www.jstor.org/stable/pdf/2346830.pdf?casa{_}token=sgr1vFuoxscAAAAA:4uYEKQ8Ww9G0-COLZUWkmr2vVR{_}-PAvY46XC0wvXVR2IzPcHj4URfheaIacMG{_}EvJfi{_}acT9RHfKEoEoxg1Q1Po-4u8jHmVe0Mge6sqaLe6qmoR84fuj.

[36] Jiyin He, Krisztian Balog, Katja Hofmann, Edgar Meij, Maarten De Rijke, Manos Tsagkias, and Wouter Weerkamp. Heuristic Ranking and Diversification of Web Documents. Technical report, 2009. URL http://ilps.science.uva.nl/.

[37] Miao He, Changrui Ren, and Han Zhang. Intent-based recommendation for B2C e-commerce platforms. *IBM Journal of Research and Development*, 2014. ISSN 0018-8646. doi: 10.1147/JRD.2014.2338091.

[38] Yuhang He and Long Chen. Fast Fashion Guided Clothing Image Retrieval: Delving Deeper into What Feature Makes Fashion. 2016. URL http://120.24.71.152/wp-content/themes/twentytwelve/pub{_}pdf/FastFashionGuidedClothingImageRetrieval.pdf.

[39] DR Hill. A vector clustering technique. *Mechanised Information Storage, Retrieval and Dissemination, North-Holland, Amsterdam*, 1968.

[40] Dietmar Jannach and Malte Ludewig. Determining characteristics of successful recommendations from log data. *Proceedings of the Symposium on Applied Computing - SAC '17*, pages 1643–1648, 2017. doi: 10.1145/3019612.3019757. URL http://dl.acm.org/citation.cfm?doid=3019612.3019757.

[41] Dietmar Jannach, Lukas Lerche, Iman Kamehkhosh, and Michael Jugovac. What recommenders recommend: an analysis of recommendation biases and possible countermeasures. *User Modeling and User-Adapted Interaction*, 25:427–491, 2015. doi: 10.1007/s11257-015-9165-3. URL https://link.springer.com/content/pdf/10.1007{%}2Fs11257-015-9165-3.pdf.

[42] Dietmar Jannach, Malte Ludewig, Lukas Lerche, and B Dietmar Jannach. Session-based item recommendation in e-commerce: on short-term intents, reminders, trends and discounts. 27:351–392, 2017. doi: 10.1007/s11257-017-9194-1. URL https://link.springer.com/content/pdf/10.1007{%}2Fs11257-017-9194-1.pdf.

[43] Yehuda (Yahoo Research) Koren, Robert (AT&T Labs-Research) Bell, and Chris (AT&T Labs-Research) Volinsky. Matrix Factorization Techniques For Recommender Systems. Technical report, 2009. URL https://datajobs.com/data-science-repo/Recommender-Systems-[Netflix].pdf.

[44] Tuomo Korenius, Jorma Laurikkala, and Martti Juhola. On principal component analysis, cosine and Euclidean measures in information retrieval. *Information Sciences*, 2007. doi: 10.1016/j.ins.2007.05.027. URL www.elsevier.com/locate/ins.

[45] Uichin Lee, Zhenyu Liu, and Junghoo Cho. Automatic identification of user goals in web search. *Proceedings of the 14th international conference on World Wide Web*, (1): 391–400, 2005. ISSN 1-59593-046-9. doi: 10.1145/1060745.1060804. URL http://dl.acm.org/citation.cfm?id=1060804.

[46] Aristidis Likas, Nikos Vlassis, and Jakob Verbeek. The global k-means clustering algorithm The global k-means clustering algorithm. [Technical. Technical report, 2003. URL https://hal.inria.fr/inria-00321515.

[47] Pablo Loyola, Chen Liu, and Yu Hirate. Modeling User Session and Intent with an Attention-based Encoder-Decoder Architecture. *Proceedings of the Eleventh ACM Conference on Recommender Systems - RecSys '17*, pages 147–151, 2017. doi: 10.1145/3109859.3109917. URL http://dl.acm.org/citation.cfm?doid=3109859.3109917.

[48] Mark Meiss, John Duncan, Bruno Gonçalves, José J Ramasco, and Filippo Menczer. What's in a Session: Tracking Individual Behavior on the Web. Technical report, 2009. URL https://arxiv.org/pdf/1003.5325.pdf.

[49] Wendy W. Moe. Buying, Searching, or Browsing: Differentiating Between Online Shoppers Using In-Store Navigational Clickstream. *Journal of Consumer Psychology*, pages 29–39, 2003.

[50] Alan L. Montgomery, Shibo Li, Kannan Srinivasan, and John C. Liechty. Modeling Online Browsing and Path Analysis Using Clickstream Data. *Marketing Science*, 23 (4):579–595, 2004. ISSN 0732-2399. doi: 10.1287/mksc.1040.0073. URL http://pubsonline.informs.org/doi/10.1287/mksc.1040.0073.

[51] T. Pham-Gia and T. L. Hung. The mean and median absolute deviations. Technical Report 7-8, 2001. URL www.elsevier.nl/locate/mcm.

[52] Gang Qian, Shamik Sural, and Yuelong Gu. Similarity between Euclidean and cosine angle distance for nearest neighbor queries. Technical report, 2004. URL http://www.cse.msu.edu/{~}pramanik/research/papers/papers/sac04.pdf.

[53] Joseph Lee Rodgers and W Alan Nicewander. Thirteen Ways to Look at the Correlation Coefficient. Technical Report 1, 1988. URL https://www.jstor.org/stable/pdf/2685263.pdf?casa{_}token=cWOOMCYAExgAAAAA:QTUthXpqMEI0L0jCa6PUzsKrxtK3KRu43PD5y0ksfXVh6iVsEepnXrLxYOPmL0-q8AWdPsyaklQnXxGCfaBEBgdoylrRCyWSZ7GDk5io4rcLMJHdWfjnwQ.

[54] Gerard Salton and Anita Wong. On the Role of Words an Phrases in Automatic Text Analysis. 1989. URL http://motore.ittig.cnr.it/EditoriaServizi/AttivitaEditoriale/InformaticaEDiritto/1976{_}03{_}291-321{_}Salton.pdf.

[55] Upendra Shardanand and Pattie Maes. Social Information Filtering: Algorithms for Automating "Word of Mouth". Technical report, 1995. URL http://ringo.media.mit.edu.

[56] Qiang Su and Lu Chen. A method for discovering clusters of e-commerce interest patterns using click-stream data. *Electronic Commerce Research and Applications*, 14:1–13, 2015. doi: 10.1016/j.elerap.2014.10.002. URL https://ac.els-cdn.com/S1567422314000726/1-s2.0-S1567422314000726-main.pdf?{_}tid=8590d3a4-109d-4423-9c58-22b31e3905e3{&}acdnat=1526462496{_}52368253c7a2dbe7f96c77cdb0ef72cb.

[57] Gábor Takács, István Pilászy, Bottyán Németh, and Domonkos Tikk. Matrix factorization and neighbor based algorithms for the netflix prize problem Matrix Factorization and Neighbor Based Algorithms for the Netflix Prize Problem General Terms. 2014. doi: 10.1145/1454008.1454049. URL https://www.researchgate.net/publication/221141015.

[58] Julián Urbano, Brian Mcfee, J Stephen Downie, and Markus Schedl. How Significant is Statistically Significant? The Case of Audio Music Similarity and Retrieval. Technical report, 2012. URL http://julian-urbano.info.